Noname manuscript No.
(will be inserted by the editor)

Estimating Simplet Counts via Sampling

Hyunju Kim* · **Heechan Moon*** · **Fanchen Bu** · **Jihoon Ko** · **Kijung Shin†**

Received: date / Accepted: date

Abstract Simplicial complexes are higher-order combinatorial structures which have been used to represent real-world complex systems. In this paper, we focus on the local patterns in simplicial complexes called *simplets*, a generalization of graphlets. We study the problem of counting simplets of a given size in a given simplicial complex. For this problem, we extend a sampling algorithm based on color coding, from graphs to simplicial complexes, with essential technical novelty. We theoretically analyze our proposed algorithm named SC3, showing its correctness, unbiasedness, convergence, and time/space complexity. Through extensive experiments on sixteen real-world datasets, we show the superiority of SC3 in terms of accuracy, speed, and scalability, compared to the baseline methods. We use the counts given by SC3 for simplicial complex analysis, especially for characterization, which is further used for simplicial complex clustering, where SC3 shows a strong ability of characterization with domain-based similarity. Additionally, we explore a variant of simplet counting (specifically, estimating the relative counts of simplets) under realistic scenarios where the entire simplicial complex is not provided at once but can only be partially accessed, for instance, through a limited number of API calls. For such scenarios, we propose a random-walk-based sampling algorithm, SCRW, and analyze its theoretical properties. In our experiments, SCRW requires, on average, 16.5× less memory than SC3, while the speed-accuracy trade-offs provided by the two methods are comparable.

Keywords Simplicial complex · Simplet · Counting algorithm · Color coding · Random walk

We sincerely thank the anonymous reviewers for their valuable comments and constructive suggestions. This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00438638, EntireDB2AI: Foundations and Software for Comprehensive Deep Representation Learning and Prediction on Entire Relational Databases, 50%) (No. 2022-0-00871 / RS-2022-II220871, Development of AI Autonomy and Knowledge Enhancement for AI Agent Collaboration, 40%) (No. RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST), 10%).

* Equal Contribution. † Corresponding Author.

H. Kim · H. Moon · J. Ko · K. Shin
Kim Jaechul Graduate School of AI, KAIST, Seoul, South Korea, 02455
E-mail: {hyunju.kim, heechan9801, jihoonko, kijungs} @ kaist.ac.kr

F. Bu
School of Electrical Engineering, KAIST, Daejeon, South Korea, 34141
E-mail: boqvezen97@kaist.ac.kr

1 Introduction

In many real-world systems, group relations involving more than two entities exist, which cannot be fully represented by pairwise graphs. For example, for co-authorship relations [55], a single publication is possibly done by more than two authors; for email systems [36], the recipients of an email can be more than two. Therefore, hypergraphs, where an edge may contain more than two nodes, naturally represent group relations involving more than two entities, and have been used to model such systems.

In spite of the representative power of hypergraphs, modeling real-world systems as hypergraphs may not be optimal in scenarios where all subset relations naturally occur within group relationships (e.g., author sets in co-authorship networks and item sets in co-purchasing

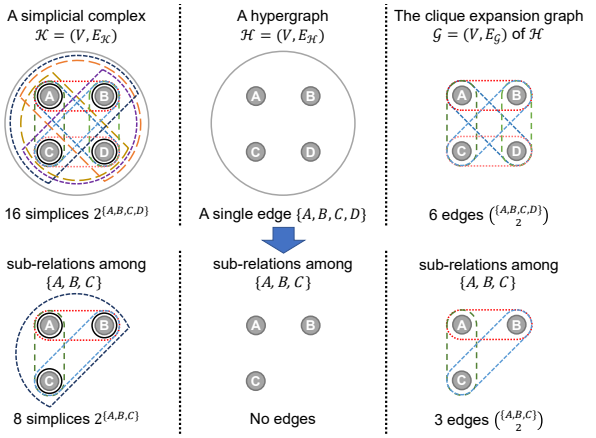


Fig. 1: A simplicial complex (left), a hypergraph (middle), and a clique expansion graph (right) representing the same group interaction among four entities. The hypergraph does not inherently represent interactions involving only subsets of the entities (which naturally exist in some real-world scenarios such as co-authorship and co-purchasing), while the clique expansion graph cannot effectively capture interactions specific to groups larger than two.

networks).[1] In hypergraphs, subset relations are often not represented as separate edges; for example, when we have a group relation involving four entities a, b, c, d, usually only a single edge $\{a, b, c, d\}$ exists in the corresponding hypergraph, as shown in Figure 1. However, this overlooks the sub-relations, e.g., the relations $\{a, b, c\}$, $\{a, b, d\}$, and $\{a, b\}$, and this makes it hard to capture local patterns. Notably, the typical pairwise-graph representation (namely, clique expansion) overlooks group interactions of size more than two.

In order to address this problem, one may use simplicial complexes (SCs) [31,43,58]. An SC is defined as a pair consisting of a set of nodes and a set of edges, where each edge is a subset of the nodes. With the *downward closure* property, for each edge in an SC, all the subsets of the edge are also included in the SC, as shown in Figure 1. Compared to hypergraph-based modeling, SC-based modeling has advantages in applications and connections to geometry and algebra [53]. Notably, in practice, we do not need additional space for the SC-based modeling (compared to the hypergraph-based one) since we can store the same set of edges as in the corresponding hypergraph with all the subset edges *implicitly* included. For example, when a group relation involves four entities: a, b, c, and d, 11 hyperedges are required to represent all possible sub-relations: $\{a, b\}$, $\{a, c\}$, $\{a, d\}$, $\{b, c\}$, $\{b, d\}$, $\{c, d\}$, $\{a, b, c\}$, $\{a, b, d\}$, $\{a, c, d\}$, $\{b, c, d\}$, and $\{a, b, c, d\}$. In contrast, SC can capture this entire group relation using just a single simplex, $\{a, b, c, d\}$. Due to these merits, SCs have been widely used to model real-world complex systems [7, 51] of communication [63], epidemic spreading [37], or social contagion [29]. See also [60] for comprehensive comparisons between SCs and (hyper)graphs.

One of the benefits of modeling real-world systems as abstract structures is that it makes studying the patterns [17] within the systems easier, especially the local patterns [34]. An important and widely-used example on pairwise graphs is graphlets [45,49] which describe the pattern of the interactions among several nodes. In typical usage of graphlets, the occurrences of each graphlet are counted in the input graph [48,42], and the counts are used to measure the similarity between graphs [54], detect anomalies [27], or detect communities [71].

In this paper, we study the problem of counting *simplets*. Simplets are the counterpart of graphlets in SCs. Similar to graphlets, simplets describe the patterns of the simplices formed by groups of nodes. The concept of simplets and the problem of counting simplets were mentioned for the first time in [47]. However, in [47], instead of directly counting the occurrences of simplets, a surrogate measure was proposed to estimate the occurrences, due to the theoretical hardness of direct counting (see Appendix B for details). Benson et al. [7] considered the problem of counting certain configurations of three or four nodes in simplicial complexes, focusing on those related to simplicial closure; however, these configurations are distinct from simplets.[2] In summary, these prior works did not directly focus on counting simplets. In this work, we aim to directly count the occurrences of simplets, filling this existing gap.

Many techniques have been proposed for counting graphlets [28,2,46,66]. Recently, graphlet-counting methods [12–14] based on color coding (CC) [3] have been proposed, especially for the graphlets of sizes more than five. For simplet counting, we propose **SC3** (**S**implet **C**ounting using **C**olor **C**oding), an algorithm using CC-based sampling, where we adapt the algorithms for graphlet counting in [12–14] with technical improvement, to deal with the intrinsic hardness of simplet counting, e.g., set enumeration and isomorphism analysis. Specifically, given an SC and a specific simplet size, in the proposed algorithm SC3, we first use

[1] Note that there can also be scenarios where assuming the existence of subset relations is not intuitive; in such cases, hypergraph modeling, which allows for the explicit inclusion or exclusion of subset relations, can be more appropriate.

[2] Their configurations do not correspond one-to-one with simplets (see Appendix B), and we empirically demonstrate that simplet counts provide stronger characterization power (see Section 7.4).

a standard CC process consisting of two steps (building and sampling) to sample candidate node sets. After that, we obtain the maximal simplices in the sub-SC induced by each candidate node set. Finally, we match each group of maximal simplices with a simplet, while taking isomorphism into consideration. We also theoretically prove the correctness, unbiasedness, convergence, and time/space complexity of SC3.

Through extensive experiments on sixteen real-world datasets, we show the empirical correctness, convergence, and high speed of SC3. Specifically, for size-4 simplets, with 100,000 samples, the counts given by SC3 have a normalized error (normalized by the ground-truth total number of simplets) lower than 5% on all the datasets. Regarding the speed, SC3 with 100,000 samples is, on average, 41× faster than an exact method for size-4 simplets.

We also use the counts of different simplets obtained by SC3 for characterizing real-world SCs. In particular, given an SC, we measure the significance of each simplet by comparing the count of the simplet in the given SC with its count in a null model (i.e., random SCs of the same size). We then construct a characteristic vector by aggregating the significance values of different simplets. We demonstrate that characteristic vectors effectively characterize SCs. Specifically, when applying k-means++ [4] to the characteristic vectors obtained using simplets of size 5 or 6 for clustering, we achieve perfect clustering (i.e., 100% clustering accuracy) of SCs based on their domains.

We further broaden the scope of simplet counting by considering real-world scenarios with restricted access, where the entire simplicial complex (SC) is not provided at once, but only a portion can be accessed. Such scenarios are common in real-world applications, such as accessing information in large online social networks via a limited number of API calls, and have been extensively studied for graphlet counting [1, 6, 8–10, 15, 16, 26, 59, 64, 65, 68]. We specifically address the problem of estimating the relative counts of simplets under restricted access. To this end, we propose SCRW, a random-walk-based sampling algorithm. The proposed method SCRW samples simplet occurrences through random walks, where random walkers only require local information (around their current position), rather than the entire SC. Based on the sampled simplet occurrences, SCRW estimates the relative count of each simplet. We present theoretical analyses of correctness, unbiasedness, convergence, and time/space complexity of SCRW. In our experiments, SCRW requires, on average, 16.5× less memory than SC3, while the speed-accuracy trade-offs provided by the two methods (SCRW and SC3) are comparable. Specifically, to achieve similar counting accuracies, SCRW requires an average of 2.9× less time than SC3 on one dataset but an average of 3.5× more on another.

In short, our contributions are five-fold:[3]

- **New problem.** To the best of our knowledge, we are the first to formulate and study the problem of directly counting simplets in a given SC, especially for simplets of sizes more than four.
- **Algorithms.** We propose SC3 for the simplet counting problem using color-coding-based sampling. We also prove the correctness, unbiasedness, convergence, and time/space complexity of SC3.
- **Accuracy of counting.** Through extensive experiments on sixteen real-world datasets, we empirically show the correctness and convergence of SC3 w.r.t. the counts of simplets, compared with several baseline methods.
- **Strong characterization power.** We use the simplet counts estimated by SC3 to characterize real-world SCs and perform clustering on them. The results demonstrate the strong characterization power of simplets.
- **Practical consideration.** We further explore realistic scenarios where the input SC is only partially accessible and propose SCRW for simplet counting in such scenarios. We demonstrate its effectiveness both theoretically and empirically.

Reproducibility: The code and data are available at `https://github.com/hhyy0401/simplet_counting`.

Roadmap: The remaining parts of the paper are as follows. In Section 2, we discuss related studies. In Section 3, we introduce basic concepts and formulate the problem of simplet counting. In Section 4, we present an accurate and fast algorithm for simplet counting and provide theoretical analysis for the algorithm. In Section 5, we introduce realistic scenarios involving restricted access and propose a new algorithm for such scenarios, with corresponding theoretical analysis. In Section 6, we suggest a characterization method for sim-

[3] This work is an extended version of our previous work [32], where we introduced a simplet sampling algorithm based on color coding [3] and characterized simplicial complexes using simplet counts. In this extended version, we further explore realistic scenarios with restricted access where only part of an input SC is accessible. We study the simplet counting problem with restricted access and propose a new algorithm SCRW for this problem with theoretical analysis (see Section 5). We expand all the experiments in our previous work to include size-6 simplets. Note that the number of size-6 simplet is 15942, which is significantly larger than that of size-4 simplets (14) and that of size-5 simplets (157) (see Sections 7.2-7.4). Finally, we evaluate the empirical effectiveness of SCRW, especially in comparison to SC3 (see Section 7.5).

Table 1: Frequently-used notations

Section	Symbol	Definition
3	$[k]$	$\{1, 2, \cdots, k-1\}$ where $k \in \mathbb{N}$
	$\mathcal{K} = (V, E)$	a simplicial complex (SC) with nodes V and edges E
	$G_{\mathcal{K}} = (V_G = V, E_G = E \cap \binom{V}{2})$	the primal graph of $\mathcal{K}$
	$M(\mathcal{K})$	the set of the maximal simplices in $\mathcal{K}$
	$\mathcal{S}^k = \{\mathcal{S}_0^k, \mathcal{S}_1^k, \ldots, \mathcal{S}_{s_{k-1}}^k\}$	the set of all s_k simplets of size k
	$\mathcal{T}^k$	the set of all treelets of size k
	$N_{oc}(\mathcal{S}; \mathcal{K})$	the absolute number of occurrences of $\mathcal{S}$ in $\mathcal{K}$
	$n_{oc}(\mathcal{S}; \mathcal{K})$	the relative number of occurrences of $\mathcal{S}$ in $\mathcal{K}$
4	$C(v, \mathcal{T}, S)$	the number of occurrences of treelet $\mathcal{T}$ colored by color set S rooted at node v
	N_{ct}	the total number of occurrences of the colorful size k treelets
	T_{ct}	the sampled occurrences of colorful treelets
	$M(\mathcal{K}[V_T])$	the set of maximal simplices in the subcomplex of $\mathcal{K}$ induced by V_T
	x	the user-defined number of samples in the sampling step

plicial complexes. In Section 7, we present experimental results. Finally, in Section 8, we conclude this work.

2 Related Work

In this section, we offer an overview of related studies.

2.1 Simplicial Complex Analysis

In this paper, we focus on a graph representation of SCs [31], where SCs are represented as higher-order networks with the downward closure property, used to describe higher-order relations in network-like structures [11]. In [7], a triangle-like structure involving three nodes called *simplicial closure* is studied on real-world SCs for higher-order link prediction. Random walks on SCs are studied in [52] for spectral embedding and estimating the importance of edges. In [5], Structural information of SCs is used for wireless network traffic analysis and discrete vector field processing. Several centrality measures are defined in [22] and applied to analyze real-world protein interaction networks. Also, SCs are used to study systems in communication [63], epidemic spreading [37], or social contagion [29]. See also [11] for a comprehensive introduction to SCs.

2.2 Local Pattern Extraction via (Generalized) Graphlets

Extracting local patterns from the abstract graph modelings is a common approach for studying real-world systems [17]. For pairwise graphs, graphlets [45, 49] have been proposed to describe the interactions among a group of nodes. The counts of graphlets are used as characteristic measures of the graph [48, 42], and further used to measure graph similarity [54], detect anomalies [27], or detect communities [71].

Recently, graphlet-like patterns have also been studied on SCs. In [7], local patterns in SCs consisting of three interconnected nodes (triangles) are studied.[4] A more comprehensive concept called *simplets* generalizing graphlets to SCs is proposed in [47]. Similar to graphlets, each simplet can be seen as a connected SC without order or node labels, or an isomorphic equivalence class. There are also several trials on extending graphlets to hypergraphs. In [35], connectivity patterns w.r.t the intersections within each group of three edges are studied, where the patterns involve edges as the objects and are limited to groups consisting of three edges only. In [38], another generalization of graphlets on hypergraphs is proposed, where only the patterns consisting of up to four nodes are considered.

2.3 Graphlet Counting and Simplet Counting

There are sophisticatedly designed efficient algorithms for exactly counting graphlets of sizes up to five [2, 46]. However, the techniques used in those algorithms are too specific for the graphlets of limited sizes, and cannot be extended for counting simplets. Recently, for counting graphlets of size more than five, in [12–14], approximate sampling-based methods based on color coding (CC) [3] have been proposed.

Apart from CC, various random walk techniques [1, 6, 8–10, 15, 16, 26, 59, 64, 65, 68] have been developed to enumerate graphlets, each employing their own state spaces and transition strategies. One common approach is utilizing a generalized graph known as the subgraph relationship graph, where each node corresponds to a connected induced subgraph (CIS), and two nodes (i.e.,

[4] Patterns with four nodes are also briefly discussed in [7].

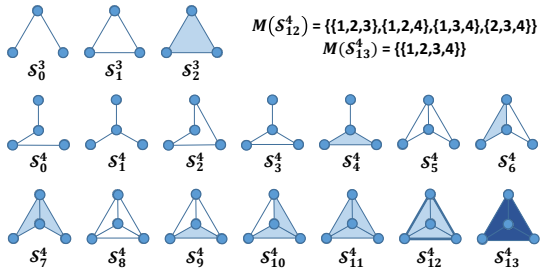


Fig. 2: All the simplets of sizes 3 and 4. An edge between two nodes indicates the presence of a simplex that contains both nodes. A light blue-filled triangle or a triangle formed by thick edges (used only for $\mathcal{S}^4_{12}$) indicates the presence of a simplex containing all three nodes in the triangle. The dark blue-filled tetrahedron (used only for $\mathcal{S}^4_{13}$) represents the presence of a simplex containing all four nodes in the tetrahedron. The detailed mathematical descriptions of $\mathcal{S}^4_{12}$ and $\mathcal{S}^4_{13}$ are provided at the top right corner.

two CISs) are connected by an edge in the subgraph relationship graph when they share a common set of nodes in the original graph. For example, GUISE [10] defines a state as a CIS of size $k-1$, k, or $k+1$, where k is the desired graphlet size, and transitions to a next state by adding, replacing, or deleting a node. PSRW [64] considers a state as a CIS with a size of $k-1$ and moves to the next state when there are $k-2$ shared nodes. SRW [15] defines a state as a CIS with a size of $2 \leq l < k-1$ (l is a hyperparameter) and moves to a next state when they have $l-1$ nodes in common. These techniques require either full access [10,64,68] or restricted access [1,6,8,9,15,16,26,59,65] to the input SC. In this paper, we focus on extending SRW with l set to 2 from graphs to SCs under restricted access.

It is even harder to count simplets due to the intrinsically more complicated nature of SCs (compared to pairwise graphs). In [47] where the concept of simplets is proposed, the authors showed the NP-hardness of simplet counting. Due to the theoretical hardness, instead of directly counting the occurrences of simplets, the authors proposed a surrogate measure called *support* and used it for indirect estimation. Essentially, only the problem of computing the surrogate measure is studied.[5] To the best of our knowledge, we are the first to study and propose a practical algorithm for the problem of directly counting the occurrences of simplets.

[5] The decision version is mainly studied in [47], and the problem of exactly computing the supports is discussed in the appendix of [47].

3 Concepts and problem statement

In this section, we introduce the main concepts used in this paper and present the formal statement of the simplet counting problem.

3.1 Concepts

Basic notations: Let $\mathbb{N}$ denote the set of positive integers. Given a set A and $k \in \mathbb{N}$, we use $2^A = \{A' : A' \subseteq A\}$ to denote the power set of A, use $[k]$ to denote $\{0, 1, \ldots, k-1\}$, and use $\binom{A}{k} = \{A' \subseteq A : |A'| = k\}$ to denote the set of all k-subsets of A.
Hypergraphs: A *hypergraph* $\mathcal{H} = (V, E)$ is defined by a node set $V = V(\mathcal{H})$ and an hyperedge set $E = E(\mathcal{H}) \subseteq 2^V$.
Simplicial complexes: A simplicial complex (SC) $\mathcal{K} = (V, E)$ is also defined by a node set V and an edge set E, while satisfying the *downward closure* property. That is, for each edge $e \in E$, all the subsets of e are also in E (i.e., $2^e \subseteq E, \forall e \in E$). In an SC, each edge e is also called a *simplex*, and an induced subcomplex on V' is the SC $\mathcal{K}[V'] = (V', E \cap 2^{V'})$. A simplex e is called a k-simplex if e has size $k+1$ (i.e., $|e| = k+1$). A simplex $e \in E$ is called a *maximal* simplex if there is no strict superset of e in $\mathcal{K}$ (i.e., $\nexists e' \in E$ s.t. $e' \supsetneq e$). We use $M(\mathcal{K})$ to denote the set of all the maximal simplices in $\mathcal{K}$. The *primal graph* $G_{\mathcal{K}}$ of $\mathcal{K}$ is a subcomplex of $\mathcal{K}$ consisting of all 1-simplices (i.e., $G_{\mathcal{K}} = (V_G = V, E_G = \{e \in E : |e| = 2\} = E \cap \binom{V}{2})$). An SC is *connected* when its primal graph is connected.
Simplets: Two SCs $\mathcal{K} = (V, E)$ and $\mathcal{K}' = (V', E')$ are *isomorphic* (denoted by $\mathcal{K} \simeq \mathcal{K}'$) if there is a bijection $\phi = \phi(\mathcal{K}, \mathcal{K}') : V \to V'$ such that $e = (v_1, v_2, \ldots, v_t) \in E$ if and only if $e' = \phi(e) \in E'$, where $\phi(e) = (\phi(v_1), \phi(v_2), \ldots, \phi(v_t))$. We also write $\mathcal{K}' = \phi(\mathcal{K})$, and ϕ is called an *isomorphic bijection* from $\mathcal{K}$ to $\mathcal{K}'$.[6]

A *simplet*, which was first introduced in [47], $\mathcal{S} = (V_{\mathcal{S}}, E_{\mathcal{S}})$ of size $k = |V_{\mathcal{S}}| \in \mathbb{N}$ is a *connected* SC, and two simplets are seen as the same one if they are isomorphic. Equivalently, each simplet can be seen as an isomorphic class. Therefore, WLOG, we assume that for each simplet of size k, its node set is $[k]$. Let $\mathcal{S}^k = \{\mathcal{S}^k_0, \mathcal{S}^k_1, \ldots, \mathcal{S}^k_{s_k-1}\}$ denote the set of all the simplets of size k, where $s_k = |\mathcal{S}^k|$ is the number of such simplets, for each $k \in \mathbb{N}$. Under the isomorphic equivalence relation, s_k takes the values 1, 1, 3, 14, 157, and 15,942 when k is 1, 2, 3, 4, 5, and 6, respectively.[7] In

[6] Multiple such bijections may exist, and we let ϕ be any of them.

[7] See Appendix C for an algorithm generating all the simplets of size k based on all the graphlets of the same size.

Algorithm 1: Brute-force enumeration of simplets

Input: (1) $\mathcal{K} = (V, E)$: the input simplicial complex;
(2) k: the considered size of simplets;
(3) $\mathcal{S}^k$: the set of all the simplets of size k
Output: $N_{oc}(\mathcal{S}_i^k; \mathcal{K}), \forall i \in [s_k]$: the count of each simplet

1 $N_{oc}(\mathcal{S}_i^k; \mathcal{K}) \leftarrow 0, \forall i \in [s_k]$ ▷ Initialization
2 **foreach** $V_k \in \binom{V}{k}$ **do**
3 **foreach** *bijection* $\phi : V_k \to [k]$ **do**
4 **if** $\phi(\mathcal{K}[V_k]) \in \mathcal{S}^k$ **then**
5 $N_{oc}(\phi(\mathcal{K}[V_k]); \mathcal{K}) \leftarrow N_{oc}(\phi(\mathcal{K}[V_k]); \mathcal{K}) + 1$
6 **return** $N_{oc}(\mathcal{S}_i^k; \mathcal{K}), \forall i \in [s_k]$

Figure 2, we list all the simplets of size 3 and 4. A *graphlet* [13,49] $\mathcal{L} = (V_\mathcal{L}, E_\mathcal{L})$ can be seen as a special simplet that is a pairwise graph (i.e., $|e| = 2, \forall e \in E_\mathcal{L}$).

Given an SC $\mathcal{K} = (V_\mathcal{K}, E_\mathcal{K})$ and a simplet $\mathcal{S}$ of size k, we say there is an *occurrence* of $\mathcal{S}$ on $X \in \binom{V}{k}$ in $\mathcal{K}$ (the event is denoted by $OC(\mathcal{S}, X; \mathcal{K})$), if the *induced* subcomplex $\mathcal{K}[X]$ is isomorphic to $\mathcal{S}$.[8] The total *number of occurrences* of $\mathcal{S}$ in $\mathcal{K}$ is

$$N_{oc}(\mathcal{S}; \mathcal{K}) = \left| \left\{ X \in \binom{V}{k} : OC(\mathcal{S}, X; \mathcal{K}) \right\} \right|.$$

For a given SC $\mathcal{K}$, the *relative count* of a size-k simplet $\mathcal{S} \in \mathcal{S}^k$, denoted as $n_{oc}(\mathcal{S}; \mathcal{K})$, is defined as the ratio between the count of the simplet and the total number of simplets, i.e., $n_{oc}(\mathcal{S}; \mathcal{K}) := \frac{N_{oc}(\mathcal{S}; \mathcal{K})}{\sum_{\mathcal{S} \in \mathcal{S}^k} N_{oc}(\mathcal{S}; \mathcal{K})}$.

Treelets and colorful treelets: Given an SC $\mathcal{K} = (V, E)$ and $k \in \mathbb{N}$, we apply a k-coloring $f : V \to [k]$, where to each $v \in V$ a color $f(v) \in [k]$ is assigned. We define *treelets* as special cases of simplets that are trees.[9] We use $\mathcal{T}^k$ to denote the set of all the size-k treelets. That is, $\mathcal{T}^3 = \{\mathcal{S}_0^3\}$, and $\mathcal{T}^4 = \{\mathcal{S}_0^4, \mathcal{S}_1^4\}$ (refer to Figure 2 for $\mathcal{S}_0^3$, $\mathcal{S}_0^4$, and $\mathcal{S}_1^4$). We say a treelet $\mathcal{T} \in \mathcal{T}^k$ is a *colorful* treelet [12] if all the k colors are used on the k nodes in $\mathcal{T}$ (i.e., $\{f(v) : v \in V_\mathcal{T}\} = [k]$).

The frequently-used notations are summarized in Table 1. In the notations, the input SC $\mathcal{K}$ can be omitted when the context is clear.

[8] In this paper, only induced subcomplexes are counted following the typical definition for graphlets [28], while non-induced ones are also counted in [47]. However, note that the counts of non-induced subcomplexes can be derived from those of induced subcomplexes by leveraging the set inclusion relationships among simplets.

[9] Note that the treelets defined on SCs are identical to those on graphs, and their occurrences are the same in an SC and its primal graph.

3.2 Problem Statement

We are now ready to present the formal statement of the problem studied in this paper, where we aim to count the occurrences of each simplet of a given size k in a given SC $\mathcal{K}$.

Problem 1 Given an SC $\mathcal{K} = (V, E)$ and the simplet size $k \in \mathbb{N}$, we aim to count the number $N_{oc}(\mathcal{S}_i^k; \mathcal{K})$ of occurrences of $\mathcal{S}_i^k$ in $\mathcal{K}$, for each $i \in [s_k]$.

Problem 1 extends the well-known and widely-studied counterpart problem of counting graphlets [39, 67, 28, 2, 50, 10, 26, 56, 64, 13, 14]. Due to the quadratic nature of graphlets (essentially, of pairwise relations), and the exponential nature of simplets (essentially, of group relations), Problem 1 is intrinsically more difficult than the counterpart on graphlets, while inheriting the #W[1]-hardness [30]. Specifically, the brute-force enumeration for graphlet counting takes $O(k^2 n^k)$ time [13]; while for simplet counting it takes $O(2^k n^k)$ time, where k is the given size of considered graphlets or simplets and n is the number of nodes in the input graph or SC. We present the enumeration process for simplet counting in Algorithm 1, where we first enumerate all the k-subsets V_k of the node set (Line 2) and all the bijections from V_k to $[k]$ (Line 3) to check which simplet the induced subcomplex on V_k corresponds to (this takes $O(2^k)$ time since a simplet of size k has $O(2^k)$ edges) and increment the counting accordingly (Lines 4 and 5), which gives the total time complexity $\binom{|V|}{k} \cdot k! \cdot 2^k = O(2^k |V|^k)$.[10]

Due to the prohibitive time complexity of the brute-force method, it is imperative to have a faster method for simplet counting. Unfortunately, the existing techniques for exact graphlet counting rely on sophisticated designs tailored for specific problems and cannot be directly extended to simplet counting. For example, Ahmed et al. [2] used combinatorial derivations of graphlet counts that leverage information such as the number of cliques and cycles; however, these derivations are specific to graphs and not directly applicable to SCs. Therefore, instead of exact counting, we aim to propose an approximate algorithm with high accuracy.

4 Proposed method: SC3

In this section, we introduce in detail our algorithm **SC3** (**S**implet **C**ounting using **C**olor **C**oding). We present the algorithmic details based on color coding (CC) [3] for counting the occurrences of each specific

[10] See [47] for more discussion on the hardness of this problem.

Algorithm 2: SC3-build

Input:
(1) k: the considered size of simplets
(2) $G_{\mathcal{K}} = (V_G, E_G)$: the primal graph of the input SC $\mathcal{K}$
Output:
(1) $C(v, \mathcal{T}, S), \forall v \in V, \mathcal{T}, S$: the number of occurrences of $\mathcal{T}$ colored by S rooted at v
(2) N_{ct}: the total count of occurrences of size-k colorful treelets

1 **foreach** $v \in V$ **do**
2 **foreach** $\mathcal{T} \in \bigcup_{i=1}^{k} \mathcal{T}^i$ **do**
3 **foreach** $S \in \binom{[k]}{|\mathcal{T}|}$ **do**
4 $C(v, \mathcal{T}, S) \leftarrow 0$
5 **foreach** $v \in V$ **do**
6 $c(v) \leftarrow$ uniformly at random sampled in $[k]$
7 $C(v, (\{0\}, \emptyset), \{c(v)\}) \leftarrow 1$
8 **for** $i = 2, \ldots, k$ **do**
9 **foreach** $v \in V$ **do**
10 **if** $i = k$ *and* $c(v) \neq 0$ **then**
11 **continue**
12 **foreach** $\mathcal{T} \in \mathcal{T}^i$ **do**
13 **foreach** $S \in \binom{[k]}{i}$ **do**
14 $C(v, \mathcal{T}, S) \leftarrow \frac{1}{d} \sum_{(u,v) \in E_G} \sum_{S_1 \sqcup S_2 = S} C(v, \mathcal{T}_1, S_1) \cdot C(u, \mathcal{T}_2, S_2)$
15 $N_{ct} \leftarrow 0$
16 **for** $v \in V$ **do**
17 **foreach** $\mathcal{T} \in \mathcal{T}^k$ **do**
18 $N_{ct} \leftarrow N_{ct} + C(v, \mathcal{T}, [k])$
19 **return** C, N_{ct}

Algorithm 3: SC3-sample

Input:
(1) k: the considered size of simplets
(2) $G_{\mathcal{K}} = (V_G, E_G)$: the primal graph of the input SC $\mathcal{K}$
(3) $C(v, \mathcal{T}, S), \forall v \in V, \mathcal{T}, S$: the number of occurrences of $\mathcal{T}$ colored by S rooted at v
(4) N_{ct}: the total count of occurrences of size-k colorful treelets
(5) x: the number of samples
Output: T_{ct}: the sampled occurrences of colorful treelets

1 **Function** Sample($v, \mathcal{T}, S$)**:**
2 **if** $|\mathcal{T}| = 1$ **then**
3 **return** $\{v\}$
4 decompose $\mathcal{T}$ with $\mathcal{T}_1$ and $\mathcal{T}_2$
5 choose $u \in N(v), S_1 \in \binom{[k]}{|\mathcal{T}_1|}, S_2 \in \binom{[k]}{|\mathcal{T}_2|}$ from distribution $\Pr(u, S_1, S_2) \propto C(v, \mathcal{T}_1, S_1) \cdot C(u, \mathcal{T}_2, S_2)$
6 **return** Sample$(v, \mathcal{T}_1, S_1) \cup$ Sample$(u, \mathcal{T}_2, S_2)$
7 $T_{ct} = \emptyset$
8 **foreach** $i \in [x]$ **do**
9 Choose $v, \mathcal{T}$ from distribution $\Pr(v, \mathcal{T}) = C(v, \mathcal{T}, [k])/N_{ct}$
10 $V_T =$ Sample$(v, \mathcal{T}, [k])$
11 $T_{ct} \leftarrow T_{ct} \cup \{V_T\}$
12 **return** T_{ct}

simplet of size k. The proposed algorithm SC3 consists of four phases: building phase, sampling phase, scanning phase, and lastly, matching phase, which extends the CC-based graphlet-counting algorithms [12–14] to simplet counting with essential technical novelty, especially in the scanning and matching phases. The theoretical analysis is provided in Section 4.5, where we show the unbiasedness, convergence, and time/space complexity of SC3.

4.1 Overview of SC3

The four steps of SC3 can be summarized as follows: for a given SC $\mathcal{K}$ and size k, **(i)** we build *trees* based on the input SC for the latter steps (building step). After that, we **(ii)** sample a (colorful and non-induced) *tree* $T = (V_T, E_T)$ such that $|V_T| = k$, uniformly at random (sampling step), **(iii)** find an induced subcomplex $\mathcal{K}[V_T]$ based on the node set of T (scanning step), **(iv)** match it to an isomorphic simplet $\mathcal{S} \in \mathcal{S}_k$ (matching step), and repeat **(ii)-(iv)** for multiple steps. The entire process is presented in Algorithm 6. See Appendix D for a toy example. Below, we present each step in detail.

4.2 Building and sampling colorful treelets

For the first two steps (building and sampling), the key techniques are adapted from color coding [3] for graphlet counting. Specifically, we use the corresponding steps of the graphlet counting algorithm in [12–14], as discussed in greater detail at the end of this subsection. Below, we elaborate on the two steps, with pseudocode provided in Algorithms 2 and 3.

Building (Algorithm 2): The primal graph $G_{\mathcal{K}}$ of an SC $\mathcal{K}$ is given as an input of the building step together with the considered size k of simplets. For each tuple $(v, \mathcal{T}, S)$ consisting of a node $v \in V$, a treelet $\mathcal{T}$ of size at most k, and a set of colors S of the same size as $\mathcal{T}$ (thus $\mathcal{T}$ colored by S is colorful), we record the number $C(v, \mathcal{T}, S)$ of occurrences of $\mathcal{T}$ colored by S rooted at v (Line 14). At the same time, we count the total number N_{ct} of occurrences of the colorful size-k treelets (Line 18).

Specifically, we first apply a k-coloring to the input $G_{\mathcal{K}}$ by coloring each node with a color in $[k]$ uniformly at random (Line 6). For counting the occurrences of color treelets, the key idea is recursively computing each $C(v, \mathcal{T}, S)$ from each pair of $C(v, \mathcal{T}_1, S_1)$ and $C(v, \mathcal{T}_2, S_2)$ with $\mathcal{T} = \mathcal{T}_1 \sqcup \mathcal{T}_2$ and $S = S_1 \sqcup S_2$ until the size of $\mathcal{T}$ reaches k, where $\sqcup$ denotes the operation of disjoint union (Lines 12-14).

Sampling (Algorithm 3): Besides the input primal graph $G_{\mathcal{K}}$ and the size k, the output of the building step (the numbers of occurrences of rooted colorful treelets $C(\cdot,\cdot,\cdot)$ and the total number N_{ct} of occurrences of the colorful treelets of size k), as well as a user-defined number x of samples to draw are given. We sample a set T_{ct} of x occurrences of colorful treelets uniformly at random among all the N_{ct} occurrences of colorful treelets of size k, where each occurrence is output as a set of nodes. The key idea is that each occurrence of a colorful treelet is sampled with a probability proportional to $C(v,\mathcal{T},[k])$, which is achieved by recursively sampling subtrees (Line 10).[11] In our implementation, we store the calculated probabilities (Line 5) to be used for sampling the next subtree. Hence, if the same subtree is sampled during the entire sampling process, we can save time by reusing the stored probabilities.

Lemma 1 *Given primal graph $G_{\mathcal{K}} = (V_G, E_G)$, $k \in \mathbb{N}$, and a user-defined number x of samples, the building and sampling steps (Algorithms 2 and 3) output T_{ct} consisting of x node sets $V_k \in V_{con}$ in $O(c^k|E_G| + xk)$ time and $O(c^k|E_G|)$ space for some absolute constant $c > 1$, where $V_{con} = \{V_k \in \binom{V_G}{k} : G_{\mathcal{K}}[V_k]$ is connected$\}$. Moreover, in each iteration of sampling, a V_k is sampled with a probability proportional to $n_{st}(G_{\mathcal{K}}[V_k])$, the number of spanning trees of $G_{\mathcal{K}}[V_k]$.*

Proof Refer to the proof in Appendix A.

As mentioned above, the building and sampling phases are based on [12–14]. Specifically, after converting an SC into its primal graph, we directly applied both steps from pairwise graph methods [12–14] to the converted graph. Note that, since a subcomplex of an SC induced by a node set is connected if and only if the corresponding induced subgraph of its primal graph is connected,[12] we can sample connected induced subcomplexes in the SC by sampling connected induced subgraphs in its primal graph. For completeness, we provide the detailed processes in Algorithms 2 and 3, and refer to [12–14] for more details.

4.3 Scanning the maximal simplices (Algorithm 4)

The building and sampling steps give us a set T_{ct} of occurrences of colorful treelets. In the scanning step, we aim to find for each occurrence $V_T \in T_{ct}$ the set of maximal simplices in the induced subcomplex of $\mathcal{K}$ on V_T, where $\mathcal{K}$ is the input SC.

Algorithm 4: SC3-scan

Input:
(1) T_{ct}: the sampled occurrences of colorful treelets
(2) $\mathcal{K} = (V, E)$: the input SC
Output: $M(\mathcal{K}[V_T]), \forall V_T \in T_{ct}$: the set of maximal simplices in the subcomplex of $\mathcal{K}$ induced by each sampled V_T

1 $M(\mathcal{K}[V_T]) \leftarrow \emptyset, \forall V_T \in T_{ct}$ ▷ Initialization
2 **foreach** $V_T \in T_{ct}$ **do**
3 **foreach** $\sigma \in M(\mathcal{K})$ *s.t.* $|V_T \cap \sigma| > 1$ **do**
4 $k_\sigma \leftarrow V_T \cap \sigma$
5 **if** $\nexists q \in M(\mathcal{K}[V_T])$ *s.t.* $k_\sigma \subseteq q$ **then**
6 $M(\mathcal{K}[V_T]) \leftarrow M(\mathcal{K}[V_T]) \cup \{k_\sigma\} \setminus \{q \in M(\mathcal{K}[V_T]) : q \subsetneq k_\sigma\}$
7 **return** $M(\mathcal{K}[V_T]), \forall V_T \in T_{ct}$

In Algorithm 4, we provide the detailed process of scanning. We first initialize the set $M(\mathcal{K}[V_T])$ of maximal simplices as empty for each sampled V_T (Line 1). Then for each V_T (Line 2), and for each maximal simplex σ intersecting with V_T with more than one node (Line 3) that is maximal in the current $M(\mathcal{K}[V_T])$ (Line 5), we add the intersection $k_\sigma = V_T \cap \sigma$ into $M(\mathcal{K}[V_T])$ while removing all the strict subsets of k_σ for computational and memory efficiency without affecting correctness (Line 6). Note that we utilize the fact that $M(\mathcal{K}[V_T]) \subset \{V_T \cap \sigma : \sigma \in M(\mathcal{K})\}$ to avoid checking all simplices. Also, note that we store only $M(\mathcal{K})$ for each SC $\mathcal{K}$.

Lemma 2 *Given T_{ct} and $\mathcal{K} = (V, E)$, Algorithm 4 correctly outputs $M(\mathcal{K}[V_T])$ for all $V_T \in T_{ct}$ in $O(|M(\mathcal{K})|\hat{M}_{ct})$ time and $O(\hat{M}_{ct})$ space, where $\hat{M}_{ct} = \sum_{V_T \in T_{ct}} |M(\mathcal{K}[V_T])|$.*

Proof Refer to the proof in Appendix A.

4.4 Matching the simplets (Algorithm 5)

After the scanning step, for each sampled occurrence V_T of a colorful treelet, we have the set $M(\mathcal{K}[V_T])$ of maximal simplices in the subcomplex induced by V_T. We aim to match each $M(\mathcal{K}[V_T])$ to a simplet $\mathcal{S} \in \mathcal{S}^k$.

Finding the isomorphic simplets is done by pre-computing the matching f for all possible permutations $\phi : [k] \to [k]$ of the nodes in each simplet of size k (Line 2). Note that the pre-computed mapping function f maps (the maximal simplices of) a subcomplex to the corresponding simplet. For each sampled $V_T \in T_{ct}$, we recover the whole SC from its maximal simplices and find the corresponding simplet by using the mapping

[11] The sampled node sets might be duplicated, and then the output T_{ct} is a multiset.

[12] Since an SC contains every 1-simplex (i.e., edge) of its primal graph, if the primal graph is connected, the SC is also connected. By the downward closure property, if two nodes are connected by a simplex of any size, they are also connected by a 1-simplex (i.e., an edge in the primal graph). Therefore, if an SC is connected, its primal graph is also connected.

Algorithm 5: SC3-match

Input:
(1) T_{ct}: the sampled occurrences of colorful treelets
(2) $M(\mathcal{K}[V_T]), \forall V_T \in T_{ct}$: the set of maximal simplices in the subcomplex induced by each sampled V_T
(3) k: the considered size of simplets/treelets
(4) $\mathcal{S}^k$: the set of all the simplets
(5) N_{ct}: the total count of the occurrences of colorful treelet
Output: $\tilde{N}_{oc}(\mathcal{S}), \forall \mathcal{S} \in \mathcal{S}^k$: the estimated count of each simplet

1 $\tilde{N}_{oc}(\mathcal{S}) \leftarrow 0, \forall \mathcal{S} \in \mathcal{S}^k$ ▷ Initialization
2 $f(\phi(M(\mathcal{S}))) \leftarrow \mathcal{S}, \forall \mathcal{S} \in \mathcal{S}^k, \forall$bijection $\phi : [k] \rightarrow [k]$
3 $n_{st}(G_{\mathcal{S}}) \leftarrow$ spanning tree count in $G_{\mathcal{S}}, \forall \mathcal{S} \in \mathcal{S}^k$ ▷ Kirchhoff's Matrix-Tree Theorem [61]
4 **foreach** $V_T \in T_{ct}$ **do**
5 $\quad \mathcal{S}_T \leftarrow f(M(\mathcal{K}[V_T]))$
6 $\quad \tilde{N}_{oc}(\mathcal{S}_T) \leftarrow \tilde{N}_{oc}(\mathcal{S}_T) + \frac{1}{n_{st}(G_{\mathcal{S}_T})} \cdot \frac{N_{ct}}{|T_{ct}|} \cdot \frac{k^k}{k!}$
7 **return** $\tilde{N}_{oc}(\mathcal{S}), \forall \mathcal{S} \in \mathcal{S}^k$

function f (Line 5), and increase the estimated count by a normalized value ($\frac{1}{n_{st}(G_{\mathcal{S}_T})} \cdot \frac{N_{ct}}{|T_{ct}|} \cdot \frac{k^k}{k!}$ on Line 6), where $n_{st}(G_{\mathcal{S}_T})$ is the number of spanning trees in the primal graph of $G_{\mathcal{S}_T}$, pre-computed by using the Kirchhoff's Matrix-Tree Theorem [61] on the primal graph of $G_{\mathcal{S}}, \forall \mathcal{S} \in \mathcal{S}^k$ (Line 3).[13] The term $n_{st}(G_{\mathcal{S}_T})$ is used because each V_T can be sampled from $n_{st}(G_{\mathcal{S}_T})$ different trees in the sampling step, the term $\frac{N_{ct}}{|T_{ct}|}$ is the proportion of the sampled occurrences, and the term $\frac{k^k}{k!}$ comes from the fact that for k-set of nodes, there are $k!$ ways of k-coloring for it to be colorful while there are k^k ways in total.

Lemma 3 *Given (i) k, the considered size of simplets, (ii) $\mathcal{S}^k$, derived from k, (iii) N_{ct}, obtained from the building step, (iv) T_{ct}, obtained from the scanning step, (v) $M(\mathcal{K}[V_T])$ ($\forall V_T \in T_{ct}$), obtained from the sampling step, Algorithm 5 takes $O(k!\hat{M}_k + |T_{ct}| + k^\omega)$ time and $O(\hat{M}_k + \hat{M}_{ct})$ space, where $\hat{M}_k = \sum_{\mathcal{S} \in \mathcal{S}^k} |M(\mathcal{S})|$, $\hat{M}_{ct} = \sum_{V_T \in T_{ct}} |M(\mathcal{K}[V_T])|$, and ω is the exponent of the time complexity of matrix multiplication. Here, $\hat{M}_k$ is a function of k: $\hat{M}_4$ is 47 and $\hat{M}_5$ is 807.*

Proof Refer to the proof in Appendix A.

[13] The theorem states that the number of spanning trees in the given graph is equivalent to any cofactor of its Laplacian matrix. A cofactor of the Laplacian matrix of size $k \times k$ can be computed using the determinant of one of its submatrices of size $(k-1) \times (k-1)$. The time complexity of calculating the determinant of a $(k-1) \times (k-1)$ matrix is $O(k^\omega)$, where ω represents the exponent of the time complexity of matrix multiplication [13].

4.5 Theoretical Analysis

Now we conclude the whole process of SC3 in Algorithm 6, and theoretically analyze its properties. Specifically, we show the following properties:

- **Unbiasedness:** the output $\tilde{N}_{oc}(\mathcal{S})$ is an unbiased estimator of the ground truth $N_{oc}(\mathcal{S})$, for each $\mathcal{S}$;
- **Convergence:** the output $\tilde{N}_{oc}(\mathcal{S})$ converges to the ground truth $N_{oc}(\mathcal{S})$, for each $\mathcal{S}$, as the number of repeated trials increases. For a single trial of SC3 (i.e., a fixed coloring), the output converges (though not necessarily to the ground truth), as the number of samples (i.e., x) in the sampling step increases.
- **Complexities:** the time and space complexities of SC3 are bounded as functions of $\mathcal{K}$, k, and x.

Theorem 1 (unbiasedness) *Given k, $\mathcal{K}$, $\mathcal{S}^k$, and any x, for each $\mathcal{S} \in \mathcal{S}^k$, the $\tilde{N}_{oc}(\mathcal{S})$ given by Algorithm 6 satisfies that $\mathbb{E}[\tilde{N}_{oc}(\mathcal{S})] = N_{oc}(\mathcal{S})$.*

Proof Refer to the proof in Appendix A.

Theorem 2 (convergence) *Given any k, $\mathcal{K} = (V, E)$, and $\mathcal{S}^k$, for each $\mathcal{S} \in \mathcal{S}^k$, let $\tilde{N}^i_{oc}(\mathcal{S})$ denote the output by Algorithm 6 in the i-th trial. For any $\epsilon, \lambda > 0$, there exists $R_t = O(-\lambda^{-2}|V|^{2k} \ln \epsilon)$ such that if $R > R_t$, then $\Pr[|\sum_{i \in [R]} \tilde{N}^i_{oc}(\mathcal{S})/R - N_{oc}(\mathcal{S})| \leq \lambda] \geq 1 - \epsilon$, for any $x \geq 1$, where x is the number of samples in the sampling step. For a single trial, $\Pr[|\tilde{N}_{oc}(\mathcal{S}) - N_{oc}(\mathcal{S}) r_{color}| \leq \lambda\sigma] \geq 1 - \frac{1}{\lambda^2}$ where $\sigma^2 = Var[\tilde{N}_{oc}(\mathcal{S})] = O(\frac{1}{x})$ (in terms of x only), and r_{color} is the ratio between the actual count of occurrences of colorful treelets corresponding to $\mathcal{S}$ and the expected count.*

Proof Refer to the proof in Appendix A.

Theorem 3 (complexities) *Given k, $\mathcal{K} = (V, E)$, $\mathcal{S}^k$, and the number of samples x, Algorithms 6 takes $O(c^k|E_G| + x|M(\mathcal{K})|^2 + k!\hat{M}_k)$ time and $O(c^k|E_G| + x|M(\mathcal{K})| + \hat{M}_k)$ space for some absolute constant $c > 1$, where $E_G = E \cap \binom{V}{2}$, and $\hat{M}_k = \sum_{\mathcal{S} \in \mathcal{S}^k} |M(\mathcal{S})|$ is a function of k.*

Proof Refer to the proof in Appendix A.

For Problem 1, high complexity w.r.t. k is inevitable since $|\mathcal{S}_k|$ increases exponentially w.r.t. k. However, SC3 is scalable w.r.t factors other than k and empirically much faster than the competitors. Note also from the theorems that increasing x (i.e., the number of samples) reduces variance but increases both the time and memory complexities.

Algorithm 6: SC3: Simplet Counting using Color Coding

Input: (1) k: the considered size of simplets
(2) $\mathcal{K}$: the input SC
(3) $\mathcal{S}^k$: the set of all the simplets
(4) x: the user-defined number of samples
Output: $\tilde{N}_{oc}(\mathcal{S}), \forall \mathcal{S} \in \mathcal{S}^k$: the estimated count of each simplet

$C, N_{ct} \leftarrow$ Alg. 2 with k and $G_{\mathcal{K}}$ ▷ Building
$T_{ct} \leftarrow$ Alg. 3 with k, $G_{\mathcal{K}}$, C, N_{ct} and x ▷ Sampling
$M(\mathcal{K}[V_T]) \leftarrow$ Alg. 4 with T_{ct} and $\mathcal{K}, \forall V_T \in T_{ct}$ ▷ Scanning
$\tilde{N}_{oc}(\mathcal{S}) \leftarrow$ Alg. 5 with T_{ct}, $M(\mathcal{K}[V_T]), \forall V_T \in T_{ct}$, k, $\mathcal{S}^k$, and $N_{ct}, \forall \mathcal{S} \in \mathcal{S}^k$ ▷ Matching
return $\tilde{N}_{oc}(\mathcal{S}), \forall \mathcal{S} \in \mathcal{S}^k$

5 Simplet Counting under restricted access

In this section, we extend the scope of simplet counting by exploring real-world scenarios with restricted access. Specifically, we consider scenarios where the entire input simplicial complex (SC) is not provided at once, but rather, we can only access a portion of it. Such scenarios are prevalent. For instance, in online social networks, users are often allowed to access only a small portion of data through a restricted number of API calls.

Under restricted access, we specifically consider the problem of estimating the relative counts of simplets. To this end, we propose SCRW, a sampling method based on random walks. Note that random walks require only local information centered on the current position of random walkers at a time. Specifically, we provide a problem definition in Section 5.1 and then describe SCRW in Section 5.2. Lastly, we provide theoretical analyses regarding the unbiasedness, convergence, and time/space complexity of SCRW in Section 5.3.

5.1 Problem Definition

The aforementioned realistic scenarios can be modeled by restricted access models [19,20], which specify how data can be accessed. In this work, we assume a restricted access model where the input SC can be accessed only through a limited number of neighborhood queries, akin to API calls in the examples of online social networks mentioned above. A *neighborhood query* for a query node $v \in V$ in $\mathcal{K} = (V, E)$ returns the set of maximal simplices incident to v, denoted by $\mathcal{Q}(v)$, i.e., $\mathcal{Q}(v) = \{\sigma \in M(\mathcal{K}) : v \in \sigma\}$.[14]

[14] In this paper, we assume that the dataset is provided in the form of a set of maximal simplices, i.e., $\bigcup_{\sigma \in M(\mathcal{K})} 2^{\sigma}$.

Under this restricted access, especially when the number of queries is restrictive, simplet counting becomes even more challenging because we are likely not able to access the entire SC or even reach every connected component. Therefore, we consider an easier version of simplet counting where **(i)** the input SC is connected and contains at least one k-simplex, where $k \geq 2$,[15] and **(ii)** we aim only to estimate the relative counts (i.e., proportions or concentrations; refer to Section 3.1) of simplets rather than their absolute counts. Formally, the problem is defined as follows.

Problem 2 Given the number of queries y on a connected SC $\mathcal{K} = (V, E)$, the simplet size $k \in \mathbb{N}$, and the node set V, we aim to compute the relative count $n_{oc}(\mathcal{S}_i^k; \mathcal{K})$ for each $i \in [s_k]$ in $\mathcal{K}$ while accessing $\mathcal{K}$ only through up to y neighborhood queries.

5.2 Proposed Method: SCRW

In this section, we present **S**implet **C**ounting using **R**andom **W**alks (SCRW), our proposed algorithm for addressing Problem 2.

5.2.1 Key Concepts for SCRW

We begin by defining several key concepts that form the foundation of SCRW.

Random walks on SCs: Several schemes of random walks on graphs have been explored for various purposes [1,6,8–10,15,16,26,59,64,65,68]. Among these schemes, we leverage a random-walk method using 1-simplices, which extends SRW(2) [15] for graphs. Such a method has the merit of general applicability to simplet counting regardless of simplet size and type. Specifically, for a given SC $\mathcal{K} = (V, E)$, we consider the state space of SCRW, $\mathcal{X}$, as the set of 1-simplices (recall that each 1-simplex contains two nodes) in E. Transitioning from the current state (i.e., a 1-simplex), the next state is selected from among the 1-simplices incident to the current one. Note that although these random walks are conceptually equivalent to random walks on the edges of the primal graph, SCRW does not explicitly construct the primal graph, whereas SC3 does.

States and transition probabilities: For each time step $i = 0, 1, 2, \ldots$, we denote $X_i \in \mathcal{X}$ as the i-th state of the random walk and its corresponding 1-simplex as e_{X_i}. We let $P \in \mathbb{R}^{|\mathcal{X}| \times |\mathcal{X}|}$ denote the transition probability matrix of the random walk, where each entry $P(e, e')$ is non-zero if and only if the two

[15] Note that an SC is connected if and only if its primal graph is connected, as proven in Footnote 12.

1-simplices e and e' are incident, i.e., $|e \cap e'| = 1$. Starting from the initial state X_0, the random walk at the state X_i transitions to the next state X_{i+1} with a probability $P(e_{X_i}, e_{X_{i+1}})$. If $k-1$ consecutive previous states $X_t, \ldots, X_{t+k-2}$ at a time step $t \geq 0$ are all distinct, a size-k (connected) subcomplex induced by $V' = \bigcup_{l=t}^{t+k-2} e_{X_l}$ is sampled, which is an instance of (i.e., isomorphic to) a simplet of size k. Note that employing the random walk that traverses adjacent 0-simplices, where the next state X_{i+1} is chosen among the adjacent nodes of X_i, overlooks (i.e., cannot sample) certain types of simplets, unless we consider longer consecutive states.[16]

Stationary distribution: Given an SC $\mathcal{K} = (V, E)$ that is (i) connected and (ii) contains at least one k-simplex, where $k \geq 2$, the random walk on its 1-simplices can be regarded as an irreducible and aperiodic Markov chain, since the next state X_{i+1} is determined solely based on the current state X_i, without being influenced by the previous states $X_1, \ldots, X_{i-1}$. Note that the condition (i) (i.e., connectedness) implies irreducibility[17] and the condition (ii) implies aperiodicity.[18] Thus, the random walk on 1-simplices converges to a unique *stationary distribution* π_P, which satisfies $\pi_P = \pi_P P$ [25]. Here, π_P is a row vector whose entries sum up to 1, where for each 1-simplex e, $\pi_P(e)$ is the probability of being at e after convergence.

Simplet coefficient: To ensure the unbiasedness of the proposed algorithm, it is necessary to consider appropriate weighting for each sampled induced subcomplex. Hence, we extend the concept of the state corresponding coefficient [15] used in graphs to SCs, and call the extended concept the *simplet coefficient*.

Definition 1 (simplet coefficient) For each simplet $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S})$ of size k (i.e., $|V_\mathcal{S}| = k$), its simplet coefficient $\alpha_\mathcal{S}$ is defined as the number of sequences consisting of $k-1$ simplices $e_1, \ldots, e_{k-1}$ that satisfy the following properties:

1. (1-simplices only) $e_j \in E_\mathcal{S}$ and $|e_j| = 2$, $\forall j \in [k-1]$
2. (incident consecutive pairs only) $|e_j \cap e_{j+1}| = 1$, $\forall j \in [k-2]$
3. (all nodes covered) $\bigcup_{j \in [k-1]} e_j = V_\mathcal{S}$

[16] For example, when $M(\mathcal{K}) = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}\}$, if a random walker traverses adjacent 0-simplices, it is impossible to visit all nodes v_1, v_2, v_3, v_4 within $k-1$, which equals 3, consecutive states. It requires at least 5 consecutive states (e.g., $v_2 \to v_1 \to v_3 \to v_1 \to v_4$).

[17] Any 1-simplex (i.e., state) can be reached from any other 1-simplex.

[18] Due to the downward closure property, a k-simplex with $k \geq 2$ implies that at least three 1-simplices (i.e., states) are mutually connected (i.e., sharing nodes), allowing the walker to return to a 1-simplex (i.e., state) in the k-simplex after any number of steps.

Table 2: The simplet coefficients when k is 4. The symbol i represents the index of simplets of size 4 in Figure 2.

i	0	1	2	3-4	5-7	8-13
$\alpha_{\mathcal{S}_i^4}$	2	6	8	10	24	48

Note that the third property, together with $|V_\mathcal{S}| = k$, implies that all the e_j's are distinct. The simplet coefficient $\alpha_\mathcal{S}$ represents the total number of random walks of length $k-1$ on the primal graph $G_\mathcal{S}$ of $\mathcal{S}$ that satisfy the three properties listed above. Due to the first property above, two simplets whose primal graphs (underlying graphlets) are identical have the same simplet coefficients. See Table 2 for the values when k is 4.

Mixing time of random walks: Denote a distribution starting from a state X_0 as $\pi^{(0)}$, which is the one-hot vector where only the entry corresponding to e_{X_0} is non-zero. Then for any $\epsilon > 0$, the mixing time $\tau(\epsilon)$ of random walks on an SC $\mathcal{K} = (V, E)$ is defined as $\tau(\epsilon) = \min\{t \geq 0 : \max_{e \in E:|e|=2} \|\pi^{(0)} P^t - \pi\|_{TV} < \epsilon\}$,[19] where P^t (i.e., P to the power of t) is the transition matrix after t steps and $\|\cdot\|_{TV}$ is a total variation distance between two distributions. Roughly speaking, this is the time needed for the state distribution to get close to the stationary distribution. In the case of SCRW, we use a pre-defined value of τ, referred to as the *time budget*, to reflect the concept of mixing time. SCRW excludes samples from the first τ steps of the random walk to approximate the stationary distribution.[20]

5.2.2 Description of SCRW

We first pre-compute a mapping f that is used to map (the maximal simplices of) a subcomplex to the corresponding simplet. Note that f is pre-computed considering all possible permutations $\phi : [k] \to [k]$ of the nodes in each simplet of size k (Line 2). Then, we start from a random initial state $X_0 \in \mathcal{X}$, i.e., a randomly chosen 1-simplex $e_{X_0} = (v_0, u_0)$, by randomly choosing $v_0 \in V$, $\sigma \in \mathcal{Q}(v_0)$ (where $\mathcal{Q}(v_0)$ is the neighborhood query of v_0 defined in Section 5.1), and $u_0 \in \sigma$ in order (Line 3-5). Then, we fetch $\mathcal{Q}(v_0)$ and store it as $\tilde{\mathcal{Q}}(v_0)$ (Line 7). Similarly, for $\mathcal{Q}(u_0)$, we store it as

[19] Since the mixing time is defined as the minimum time at which the difference is smaller than ϵ for 'every' state, we compare ϵ with the maximum over the differences at the states (i.e., 1-simplices).

[20] While this choice does not fully guarantee a close approach to the stationary distribution, it is supported by the experiments in Appendix H, where we demonstrate that the error due to sampling decreases more significantly within the first 20 steps, compared to later steps.

Algorithm 7: SCRW: **S**implet **C**ounting using **R**andom **W**alks

Input: (1) k: the considered size of simplets
(2) $\mathcal{S}^k$: the set of all the simplets of size k
(3) V: the set of nodes in the input SC $\mathcal{K}$
(4) y: the maximum number of queries on $\mathcal{K}$
(5) τ: the user-defined time budget
Output: $\tilde{n}_{oc}(\mathcal{S}), \forall \mathcal{S} \in \mathcal{S}^k$: the estimated relative count of each simplet

1 $\tilde{n}_{oc}(\mathcal{S}) \leftarrow 0, \forall \mathcal{S} \in \mathcal{S}^k$
2 $f(\phi(M(\mathcal{S}))) \leftarrow \mathcal{S}, \forall \mathcal{S} \in \mathcal{S}^k, \forall$bijection $\phi : [k] \to [k]$
3 $v_0 \leftarrow$ randomly chosen node from V
4 $\sigma \leftarrow$ randomly chosen simplex from $\mathcal{Q}(v_0)$
5 $u_0 \leftarrow$ randomly chosen node from σ
6 $e_{X_0} = (v_0, u_0)$ ▷ Initial state
7 $\tilde{\mathcal{Q}}(v_0) \leftarrow \mathcal{Q}(v_0)$
8 **for** $i = 1, \ldots, y-2$ **do**
9 $\quad \tilde{\mathcal{Q}}(u_{i-1}) \leftarrow \mathcal{Q}(u_{i-1})$ ▷ Note that, by definition of v_i, $\tilde{\mathcal{Q}}(v_{i-1}) = \tilde{\mathcal{Q}}(v_{i-2})$ or $\tilde{\mathcal{Q}}(u_{i-2})$ for $i > 1$
10 $\quad \mathtt{Nb}(e_{X_{i-1}}) \leftarrow$ `get-1-simplices`$(v_{i-1}, \tilde{\mathcal{Q}}(v_{i-1})) \cup$ `get-1-simplices`$(u_{i-1}, \tilde{\mathcal{Q}}(u_{i-1})) \setminus \{(v_{i-1}, u_{i-1}), (u_{i-1}, v_{i-1})\}$
11 $\quad \mathtt{pb}(X_{i-1}, X_i) \leftarrow \frac{1}{|\mathtt{Nb}(e_{X_{i-1}})|}$
12 $\quad e_{X_i} = (v_i, u_i) \leftarrow$ uniformly at random sampled from $\mathtt{Nb}(e_{X_{i-1}})$ where $v_i = e_{X_{i-1}} \cap e_{X_i}$
13 $\quad$ **if** $i > \tau$ **then**
14 $\quad\quad V_S \leftarrow \bigcup_{l=i-k+2}^{i} e_{X_l}$
15 $\quad\quad$ **if** $|V_S| = k$ **then**
16 $\quad\quad\quad \tilde{p} \leftarrow |\mathtt{Nb}(e_{X_{i-k+2}})| \cdot \prod_{l=i-k+3}^{i} \mathtt{pb}(X_{l-1}, X_l)$
17 $\quad\quad\quad \mathcal{S} \leftarrow f(M(\mathcal{K}[V_S]))$
18 $\quad\quad\quad \tilde{n}_{oc}(\mathcal{S}) \leftarrow \tilde{n}_{oc}(\mathcal{S}) + \frac{1}{\alpha_{\mathcal{S}} \cdot \tilde{p}}$
19 $\mathcal{N} \leftarrow \sum_{\mathcal{S} \in \mathcal{S}^k} \tilde{n}_{oc}(\mathcal{S})$
20 **for** $\mathcal{S} \in \mathcal{S}^k$ **do**
21 $\quad \tilde{n}_{oc}(\mathcal{S}) \leftarrow \frac{\tilde{n}_{oc}(\mathcal{S})}{\mathcal{N}}$
22 **return** $\tilde{n}_{oc}(\mathcal{S}), \forall \mathcal{S} \in \mathcal{S}^k$

23 **Procedure** `get-1-simplices`$(v, \tilde{\mathcal{Q}}(v))$:
24 $\quad$ **return** $\bigcup_{\sigma \in \tilde{\mathcal{Q}}(v)} \{(v, v') : v' \in \sigma \setminus \{v\}\}$

$\tilde{\mathcal{Q}}(u_0)$ (Line 9). Here, $\mathtt{Nb}(e_{X_0})$ is defined as the set of 1-simplices that share only one node in common with e_{X_0}, i.e. $\mathtt{Nb}(e_{X_0}) = \{e \in E : |e \cap e_{X_0}| = 1\}$. This is obtained by examining every 1-simplex in each maximal simplex $\sigma \in \tilde{\mathcal{Q}}(v_0) \cup \tilde{\mathcal{Q}}(u_0)$ (Line 10). Finally, the next state X_1 is determined by choosing one among $\mathtt{Nb}(e_{X_0})$ uniformly at random (Lines 12, 23, and 24).

Similarly, each i-th state X_i is determined from the previous state X_{i-1}. Note that in each i-th step for $i > 1$, we only need to call the neighborhood query once (Line 9), this is because $e_{X_{i-1}}$ and e_{X_i} share a common node, whose neighborhood query has been answered in the previous step (i.e., the $(i-1)$-th step). We repeat the above process for $y-2$ steps so that y neighborhood queries are answered in total.

While repeating this process, if at each time step i, $k-1$ consecutive states $\overrightarrow{V} = (X_{i-k+2}, \ldots, X_i)$ form a set of k distinct nodes V_S (Line 14), we conduct the scanning step on this sample as in SC3. If V_S does not contain k distinct nodes, with $\overrightarrow{V}$ not having $k-1$ distinct states (Line 15), this invalid sample is excluded from updating $\tilde{n}_{oc}$.[21] To ensure the unbiasedness, we calculate the value $\tilde{p}$ (Line 18) by multiplying $|\mathtt{Nb}(e_{X_{i-k+2}})|$ and transition probabilities (previously calculated in Line 11). This is because $|\mathtt{Nb}(e_{X_{i-k+2}})|$ and $\tilde{p}$ are proportional to the stationary distribution and the probability of the consecutive sequence $\overrightarrow{V}$ being sampled, respectively (refer to Lemma 4 below). Particularly, $|\mathtt{Nb}(e_{X_{i-k+2}})|$ can be regarded as weighting, assigning a value to each state based on the probability of being in the state (after mixing time, i.e, once the state distribution has sufficiently converged to the stationary distribution).

After finding the simplet $\mathcal{S}$ isomorphic to $\mathcal{K}[V_S]$ by using the pre-computed mapping function f (Line 17), we increase the estimated relative count by the inverse of $\alpha_{\mathcal{S}} \cdot \tilde{p}$ (Line 18), since each V_S can be sampled in α_S different ways, where $\alpha_{\mathcal{S}}$ is the simplet coefficient of $\mathcal{S}$ defined in Section 5.2.1. Note that the total number of induced subcomplexes is always bounded by $y - \tau - 2$ since a validated sample is produced only after reaching τ steps, where τ is the user-defined time budget. (Line 13). Finally, we obtain the estimated relative count of each simplet by normalizing it (Line 21).

Remarks: If an appropriate global coefficient is given, we can estimate the absolute counts instead of the relative counts. Let M be the global coefficient defined as $\sum_{e \in E: |e|=2} |\mathtt{Nb}(e)|$, which is equal to $2 \cdot |\bigcup_{\sigma \in M(\mathcal{K})} \binom{\sigma}{2}|$, i.e., twice the number of 1-simplices in $\mathcal{K}$.

If we replace $|\mathtt{Nb}(e_{X_{i-k+2}})|$ with $\frac{|\mathtt{Nb}(e_{X_{i-k+2}})|}{M}$ on Line 16 and skip the normalization step on Line 21, we obtain the estimated absolute count of each simplet as the final output, not its relative count. However, the value M is challenging to obtain or even estimate, especially when access to the entire SC is restricted.

5.3 Theoretical Analysis

We present the whole process of SCRW in Algorithm 7. Below, we theoretically analyze the properties of SCRW, focusing on the following properties:

- **Unbiasedness:** the output $\tilde{n}_{oc}(\mathcal{S})$ is an unbiased estimator of the ground truth $n_{oc}(\mathcal{S})$, for each $\mathcal{S}$;
- **Convergence:** the output $\tilde{n}_{oc}(\mathcal{S})$ converges to the ground truth $n_{oc}(\mathcal{S})$, for each $\mathcal{S}$, as the number of queries y increases;

[21] Empirically, the average ratios of invalid samples across the datasets considered in Section 7 when k is 4, 5, and 6 are less than 5%.

– **Complexities:** the time complexity is bounded as a function of $\mathcal{K}$, k, and y, while the space complexity is independent of y.

Throughout our analysis, we assume that the user-defined time budget τ is large enough, such that the stationary distribution is (asymptotically) reached.

We begin with presenting the stationary distribution and the transition probability of SCRW in Lemma 4, which is used to show the correctness and unbiasedness of SCRW.

Lemma 4 *For a given SC $\mathcal{K} = (V, E)$, the state space of SCRW corresponds to the set of 1-simplices in E, i.e., $\mathcal{X} = \{e \in E : |e| = 2\}$. The stationary distribution of each state $X \in \mathcal{X}$ is uniquely determined as $\frac{|\mathcal{N}\mathcal{b}(e_X)|}{M}$ (defined as Line 11). The transition probability from X to X' is $\frac{1}{|\mathcal{N}\mathcal{b}(e_X)|}$.*

Proof Refer to the proof in Appendix A.

Theorem 4 (unbiasedness and convergence) *Given k, $\mathcal{K}$, $\mathcal{S}^k$, for any $\mathcal{S} \in \mathcal{S}^k$ and $y > 0$, the $\tilde{n}_{oc}(\mathcal{S})$ given by Algorithm 7 satisfies $\mathbb{E}[\tilde{n}_{oc}(\mathcal{S})] = n_{oc}(\mathcal{S})$ and $Pr[|\tilde{n}_{oc}(\mathcal{S}) - n_{oc}(\mathcal{S})| \leq \lambda\sigma] \geq 1 - \frac{1}{\lambda^2}$ for any $\lambda > 0$, where $\sigma^2 = \mathrm{Var}[\tilde{n}_{oc}(\mathcal{S})]$.*

Proof Refer to the proof in Appendix A.

By the results in [15] (see Theorem 3 in [15]), for any $0 < \delta < 1$, there exists a constant ξ such that $\Pr[(1-\epsilon)n_{oc}(\mathcal{S}) \leq \hat{g} \leq (1+\epsilon)g] > 1 - \delta$ when the sample size $n > O(\xi \cdot \tau(\epsilon = 1/8))$, where g is the true count of the object, $\hat{g}$ is is estimated value, and $\tau(\epsilon = 1/8)$ is the mixing time at $\epsilon = 1/8$. This induces that for each simplet $\mathcal{S} \in \mathcal{S}^k$ and for any $0 < \delta < 1$, $\Pr[|\tilde{n}_{oc}(\mathcal{S}) - n_{oc}(\mathcal{S})| \leq \epsilon \cdot n_{oc}(\mathcal{S})] > 1 - \delta$ holds whenever the number of validated samples is bigger than $\Theta(\tau(\epsilon = 1/8))$. This implies that the required number of samples to guarantee the same accuracy is linear in $\tau(\epsilon = 1/8)$.

Theorem 5 (complexities) *The space complexity of SCRW is $O(\mathcal{Q}_{\max} + \hat{M}_k)$, where $\mathcal{Q}_{\max} = \max_{v \in V} \sum_{\sigma \in \mathcal{Q}(v)} |\sigma|$ is determined by $\mathcal{K}$, and $\hat{M}_k = \sum_{\mathcal{S} \in \mathcal{S}^k} |M(\mathcal{S})|$ is a function of k. The time complexity of SCRW is $O(y|M(\mathcal{K})|^2 + k!\hat{M}_k + yk + y\mathcal{Q}_{\max})$, where y is the maximum number of queries used in Algorithm 7 under the assumption each neighborhood query with query node $v \in V$ takes $O(\sum_{\sigma \in \mathcal{Q}(v)} |\sigma|)$ time.*

Proof Refer to the proof in Appendix A.

When comparing space complexities, SCRW outperforms SC3 in memory usage since it does not need additional space for pre-processing.

5.4 Limitations of SCRW

Similar to many graphlet-counting methods designed for partially accessible graphs [10,64,26,15,65,68], SCRW has two primary limitations, especially when compared to SC3. First, since SCRW relies on local information without any global information, it is able to estimate only the relative counts of simplets, not their absolute counts. Second, for its theoretical properties to hold, SCRW requires the input SC to be connected so that it can potentially reach every part of the SC.

Due to the second limitation, in our experiments in Section 7.5, SCRW is applied to the largest connected component (LCC)[22] rather than the entire SC for datasets that are not connected. This is equivalent to making random walks start from a 1-simplex in the LCC. Fortunately, due to the existence of giant connected components [21] in real-world datasets, we observe that the simplet concentration difference between the LCC and the entire SC is negligible (refer to Appendix G for details).

6 Characterization of simplicial complexes using simplet counts

In this section, we show (1) the counts given by SC3 are accurate in that they are close to the ground-truth counts, and (2) the counts given by SC3 can be used for characterizing SCs.

Characteristic Profile (CP): We use a measure called *characteristic profile* [44] (CP). Given a simplet size $k \in \mathbb{N}$ and an SC $\mathcal{K}$, CP measures the significance of each simplet $\mathcal{S}_i^k \in \mathcal{S}^k$ ($i \in [s_k]$) in the given SC $\mathcal{K}$. With the relative count of every simplet that we obtained, we define the *significance vector* $\mu = \mu(\mathcal{K}) = (\mu_0, \mu_1, \ldots, \mu_{s_k - 1}) \in \mathbb{R}^{s_k}$ of $\mathcal{K}$ by

$$\mu_i = \frac{\tilde{n}_{oc}(\mathcal{S}_i^k; \mathcal{K}) - \tilde{n}_{oc}(\mathcal{S}_i^k; \mathcal{K}_{\mathcal{R}})}{\tilde{n}_{oc}(\mathcal{S}_i^k; \mathcal{K}) + \tilde{n}_{oc}(\mathcal{S}_i^k; \mathcal{K}_{\mathcal{R}}) + \epsilon}, \tag{1}$$

where $\mathcal{K}_{\mathcal{R}}$ is any random SC generated by a null model (we will define the null model that we use later) from $\mathcal{K}$, and $\epsilon > 0$ is a small enough constant. In our experiments, we set ϵ to 10^{-3}. Based on the significance vector, we compute the CP of $\mathcal{K}$ as a normalized significance vector with each entry being

$$CP_i = \frac{\mu_i}{\sqrt{\sum_{i \in [s_k]} {\mu_i}^2}}.$$

[22] The largest connected component of the input SC is its largest connected subcomplex of it. Note that an SC is connected if and only if its primal graph is connected, as proven in Footnote 12.

The CP of an SC contains information on local patterns and allows us to compare multiple SCs, even when they have different sizes.

Null model: As mentioned above, a null model is required to compute the significance vector and CP of an SC. Regarding the choice of the null model, we aim to preserve the number of simplices and the size of each simplex. Given an SC $\mathcal{K} = (V, E)$, we extract all its maximal simplices. First, we randomly choose a size t with probability $|\{e \in E : |e| = t\}|/|E|$. Then, we choose a pair of maximal simplices of size t uniformly at random among all the pairs of size-t maximal simplices. Finally, we repeatedly switch nodes independently between the pair for $\lfloor i/2 \rfloor$ times. We obtain all the maximal simplices of the random SC $\mathcal{K}_{\mathcal{R}}$ by repeating the above procedure $c_{shuffle}|E|$ times and expanding all the maximal simplices. Based on our observations in Appendix E, which show that $c_{shuffle}$ set to 1,000 is sufficient for rewiring an SC, we use $c_{shuffle}$ set to 1,000 in our experiments.

7 Experiments

We performed experiments on sixteen real-world SCs using SC3 and several baseline methods, aiming to answer the following questions:

- **Q1. Accuracy:** How accurate are the counts of simplets obtained by SC3? How well do the counts converge to the ground truth values, as the number of samples increases?
- **Q2. Scalability and speed:** How fast is SC3 compared to the baseline algorithms? How does the running time of SC3 grow as the number of samples increases?
- **Q3. Characterization power across domains:** How well does the characteristic profile obtained from the counts by SC3 cluster the real-world SCs from different domains?
- **Q4. Simplet counting under restricted access:** How accurately and rapidly do the relative counts of simplets obtained by SCRW converge?

7.1 Experimental Setting

Machine: We performed all the experiments on a machine with a 3.7GHz Intel i5-9600K CPU and 64GB of memory.

Dataset: We used 16 real-world SC datasets. We provide the basic statistics of the datasets and their largest connected components in Table 3.

Competitors: We compare SC3 with two existing algorithms designed for SCs: (1) B-Exact [7] and (2) FreSCo [47]. Below, we provide a concise overview of them with a brief analysis of their time complexity. Note that B-Exact has not been extended for $k \geq 5$, and extending it is non-trivial.

Table 3: Some basic statistics of the real-world datasets. The second row in each cell presents statistics from the largest connected component (LCC) of the corresponding dataset.

Dataset $\mathcal{K} = (V, E)$	**Abbrev.**	$\lvert V\rvert$	$\lvert M(\mathcal{K})\rvert$
coauth-DBLP [7]	cD	1,924,991	1,730,664
-LCC		1,654,109	1,563,050
coauth-MAG-Geology [55,7]	cMG	1,256,385	925,027
-LCC		898,648	681,954
coauth-MAG-History [55,7]	cMH	1,014,734	774,495
-LCC		219,435	130,579
congress-bills [7,23,24]	cb	1,718	48,898
-LCC		1,718	48,898
contact-high-school [7,40]	chs	327	4,862
-LCC		327	4,862
contact-primary-school [7,57]	cps	242	8,010
-LCC		242	8,010
DAWN [7]	D	2,558	72,421
-LCC		2,290	72,153
email-Eu [7,70,36]	eEu	979	8,083
-LCC		979	8,083
email-Enron [7]	eEn	143	433
-LCC		143	433
NDC-classes [7]	Nc	1,161	563
-LCC		628	324
NDC-substances [7]	Ns	5,311	6,555
-LCC		3,065	4,533
tags-ask-ubuntu [7]	taau	3,029	95,639
-LCC		3,021	95,631
tags-stack-overflow [7]	taso	49,931	3,781,514
-LCC		49,931	3,781,514
threads-ask-ubuntu [7]	thau	125,602	149,025
-LCC		82,075	109,292
threads-math-sx [7]	thms	176,445	519,573
-LCC		152,702	496,379
threads-stack-overflow [7]	thso	2,675,955	8,694,667
-LCC		2,301,070	8,330,001

- **B-Exact** exactly counts 3- or 4-node configurations using combinatorial methods. Notably, each simplet may correspond to zero, one, or multiple node configurations of the same size. See Appendix B for detailed correspondence relations between simplets and node configurations. Regarding time complexity, the dominant step of B-Exact is to enumerate all 4-cliques in the primal graph using the Chiba and Nishizeki algorithm [18]. For a given SC $\mathcal{K} = (V, E)$, it takes $O(\alpha(\mathcal{K}) \cdot |E_1|)$ time *per clique*, where $E_1 = \{e \in E : |e| = 2\}$ is the set of 1-simplices of $\mathcal{K}$ and $\alpha(\mathcal{K}) \leq \lceil \sqrt{2|E_1| + |V|}/2 \rceil$. Therefore, the total time complexity becomes $O(\alpha(\mathcal{K}) \cdot |E_1| \cdot |K_4|)$, where K_4 is the set of 4-cliques in the primal graph of $\mathcal{K}$.

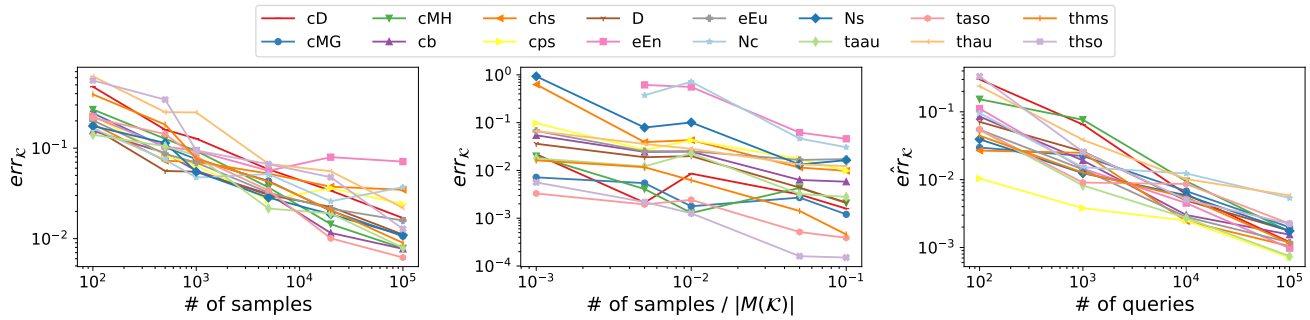


Fig. 3: The error of the proposed methods, SC3 and SCRW, shows a clear tendency to decrease as the number of samples or queries increases. We present the normalized error, $err_{\mathcal{K}}$, of SC3 with several fixed sample sizes (left figure) and with sample sizes proportional to the dataset size (center figure). We also present the normalized error, $\hat{err}_{\mathcal{K}}$, of SCRW with varying query sizes (right figure). The means of the errors over 5 trials on each dataset is reported.

– **FreSCo** indirectly estimates the count of each simplet using a surrogate measure called *support* (see Appendix B or [47] for the formal definition). The support satisfies that if two simplices $\sigma_1 \subseteq \sigma_2$ then the support of σ_1 is at least that of σ_2. We compare SC3 with FreSCo that exactly computes the support of each simplet. See Appendix B (esp. Table 5) for comparisons between the exact or estimated counts and the supports from FreSCo, where we observe that the supports computed in 10 hours (the time limit that we set) are not strongly related to the exact counts. The time complexity of FreSCo depends on the time complexity of checking isomorphism between a simplet and subcomplexes. The number of simplets with k nodes is bounded by $2^{\sum_{i=2}^{k}\binom{k}{i}}$, and determining the frequency of a simplet with k nodes involves checking all its isomorphisms, requiring up to $O(|V|^k)$. Hence, the total time complexity of FreSCo becomes $O\left(2^{\sum_{i=2}^{k}\binom{k}{i}}|V|^k\right)$ [47], which becomes $O\left(|V|^4\right)$ when k is fixed to 4.

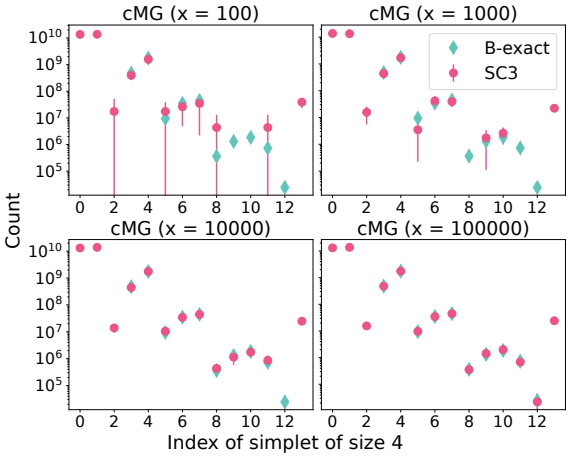


Fig. 4: The estimated counts by SC3 are closer to the exact counts as the number of samples increases. When k is 4, we report the mean value of each the counts of each simplet as a point, and the standard deviation as an error bar.

Implementations: We implemented SC3 and SCRW in C++. For SC3, we adopted multi-threading for the building and scanning steps. For SCRW, we adopted multi-threading for all steps. We consistently used six threads for all methods. We set the time budget τ to 20 based on empirical observations that this was sufficient across all datasets. Refer to Appendix H for experimental results with other time budgets.

For B-Exact, we used the open-source implementation in Julia provided by the authors. Note that Julia uses LLVM as a compiler, which is written in C++. For FreSCo, we used the open-source implementation in Java provided by the authors. For fairness, we must note potential differences in the efficiency of programming languages. Nonetheless, our primary contribution remains significant, as further discussed in Appendix F.

7.2 Q1. Accuracy

To evaluate the accuracy, we compare the ground truth counts computed by B-Exact and the estimated counts obtained by SC3 while varying the number of samples from 100 to 100,000. That is, we measured the error based only on the counts of simplets considered by both B-Exact and SC3, whose ground-truth counts can be computed by B-Exact. For the error measure,

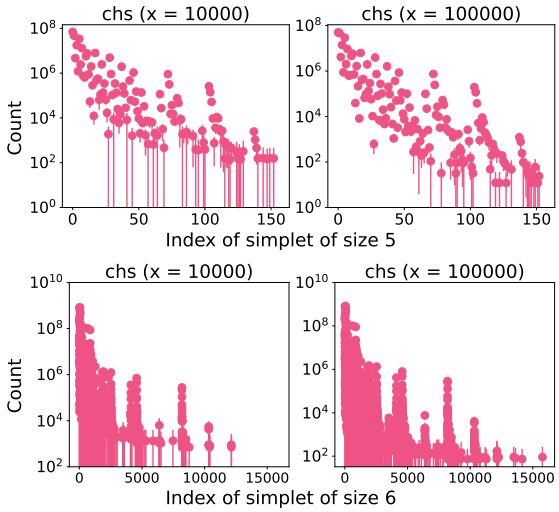


Fig. 5: The estimated counts converge as the number of samples increases. When k is 5 and 6, we computed the estimated counts on the `chs` dataset when the number of samples, x, is 10,000 (left) and 100,000 (right).

we used a normalized error $err_{\mathcal{K}}$ defined as

$$err_{\mathcal{K}} = \frac{\sum_{i \in [s_k]} |N_{oc}(\mathcal{S}_i^k; \mathcal{K}) - \tilde{N}_{oc}(\mathcal{S}_i^k; \mathcal{K})|}{\sum_{i \in [s_k]} N_{oc}(\mathcal{S}_i^k; \mathcal{K})}.$$

Since B-Exact only provides the exact counts when $k \leq 4$, we fixed k to 4. As seen in Figure 3 (spec., the left and center subfigures), the count of each $\mathcal{S}_i^k \in \mathcal{S}^k$ approaches the exact count, as the number of samples increases. Specifically, on 15 out of 16 datasets, the normalized error $err_{\mathcal{K}}$ is below 0.05, showing the high accuracy of SC3 on real-world datasets.

In addition, we visualized how the count of each simplet converges to the exact count as the number of samples increases on the `cMG` dataset. As shown in Figure 4, the count of each simplet successfully converges to the actual count, and the standard deviation of the estimation for each simplet also decreases, as the number of samples increases. We additionally computed the mean values and the standard deviations of the estimated counts when k is 5 and 6 on the `chs` datasets. As seen in Figure 5, the count of each simplet converges when the actual count is large enough, e.g., when we set the number of samples to 10,000 and 100,000.

7.3 Q2. Scalability and Speed

We compare the running times of the considered algorithms. Specifically, we measured the running time of SC3 with the number of samples $x \in \{10^3, 10^4, 10^5\}$. For the baseline algorithms, we used B-Exact for k set to 4 and FreSCo for k set to 4, 5 and 6. As seen in Figure 6, SC3 is fastest on 10 (out of 16 datasets) and is second fastest on the others. On average, SC3 is 41× faster than B-Exact. Furthermore, to alleviate the issue arising from baselines implemented in different languages, in Appendix F, we compare the growth rate of running times relative to that of dataset sizes, which is independent of absolute running times. The results in Appendix F indicate that, in most cases, SC3 exhibits a lower relative growth rate compared to the baseline methods, demonstrating its scalability.

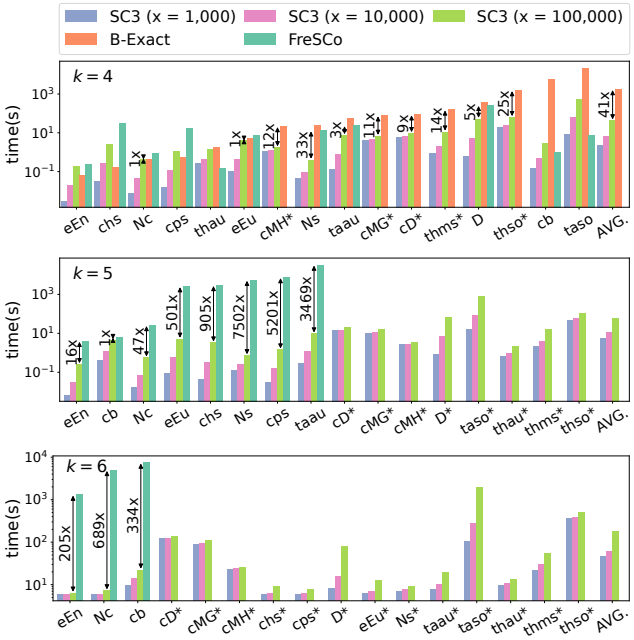


Fig. 6: SC3 is significantly faster than the baseline algorithms. FreSCo ran out of time (> 10 hours) for some datasets, and B-Exact can be used only for $k \leq 4$.

To evaluate the scalability, we additionally measured the running time of each step of SC3 on the `cD`, `taso`, and `thso` datasets. We measured the running times with different numbers of samples. As shown in Figure 7, the running times of sampling, scanning, and matching steps increase sub-linearly as the number of samples increases. Note that the running time of the building step is constant and does not depend on the number of samples.

7.4 Q3. Characterization Power across Domains

To analyze the characterization power of the counts of simplets obtained by SC3, we computed the characteristic profile (CP) for each dataset (refer to Section 6 for its definition). We first analyzed SCs within the same domain using CPs and demonstrated them in Figure 8.

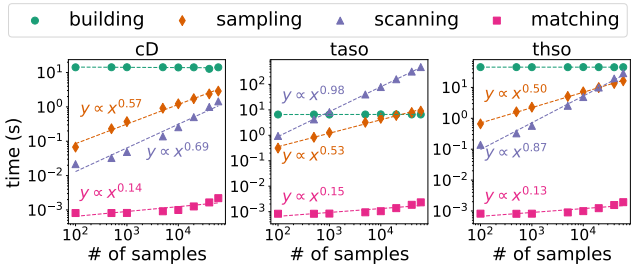


Fig. 7: Each step of SC3 is scalable. The running times of sampling, scanning, and matching steps increase sublinearly as the number of samples increases.

Table 4: The result of k-means++ (i.e., the assignment of each dataset in clusters C1 and C5) for 10 trials. When k is 5 or 6, it always succeeded in distinguishing domains, but when k is 4, it succeeded only once in ten trials.

k	coauth-*			threads-*			contact-*		email-*		tags-*		trials
	cD	cMG	cMH	thau	thms	thso	cps	chs	eEn	eEu	taau	taso	
6	C1	C1	C1	C2	C2	C2	C3	C3	C4	C4	C5	C5	10/10
5	C1	C1	C1	C2	C2	C2	C3	C3	C4	C4	C5	C5	10/10
4	C1	C1	C1	C2	C2	C3	C4	C4	C5	C5	C5	C5	9/10
	C1	C1	C1	C2	C2	C2	C3	C3	C4	C4	C5	C5	1/10

Compared to the random SCs generated by the null model, the original SCs in the `coauth-*` domain (`cD`, `cMG`, `cMG`) commonly contain more $\mathcal{S}_3^4$ and $\mathcal{S}_7^4$ with noticeable differences. In the `threads-*` domain (`thau`, `thms`, `thso`), two simplets $\mathcal{S}_3^4$ and $\mathcal{S}_5^4$, which do not contain any 2-simplex, are less in the original SCs, compared to the random SCs generated by the null model. For the `contact-*` domain (`chs`, `cps`), the numbers of $\mathcal{S}_8^4$ (4-clique) in the real-world SCs are much higher than those in the random SCs generated by the null model. In the `email-*` domain (`eEu`, `eEn`), two simplets $\mathcal{S}_5^4$ and $\mathcal{S}_8^4$, which do not contain any 2-simplex, are fewer, while $\mathcal{S}_4^4$ and $\mathcal{S}_7^4$, which contain one or more 2-simplices, are more frequent, compared to the corresponding random SCs generated by the null model.

To further evaluate the characteristic power across domains, we obtained CP vectors for the considered algorithms. Let CP^4, CP^5, CP^6, CP^B, and CP^F denote the CPs obtained from SC3 when k is 4, 5, 6, B-Exact when k is 4, and FreSCo when k is 4, respectively. For each type of CP, we computed the cosine similarity between each pair of the datasets. Specifically, for CP_B, we computed CP using 4-node configurations described in the original paper instead of simplets; for CP_F, we used the lowest $\theta \in \{10^5, 10^4, 10^3, 10^2, 10, 1\}$ such that FreSCo terminates in 10 hours.[23] Since FreSCo computes the supports instead of the counts of simplets, we computed CP^F using the ratio of the supports (instead of the counts used for SC3 and B-Exact).

As seen in Figures 9(a)-9(c), CP^4, CP^5, and CP^6 show strong characterization power, and the superiority of SC3 w.r.t the characterization power becomes clearer when k is 5 and 6. Both competitors failed to clearly distinguish the SCs in different domains, which indicates their poor characterization power, and were unable to run on certain datasets, as shown in Figures 9d and 9e.

[23] Here, θ serves as the minimum support for FreSCo, and lowering its value results in computing the support for more simplets with smaller support values.

Specifically, we suspect that FreSCo has low characterization power because support, used as a surrogate measure, offers limited information on counts. Additionally, B-Exact does not capture certain simplets, such as trees, which may impact its characterization power.

We further embedded the CP (refer to Section 6 for its definition) of each SC in Euclidean space. That is, we treated the CP from each SC, which is a vector, as the vector representation of the SC in Euclidean space. Then, we performed k-means++ clustering [4] for 10 independent trials.

In Table 4, we report the clustering results using the CP computed from the counts obtained by SC3, where the clustering results were perfect when k is 5 and 6, while for k equals 4, it often failed to cluster the SCs properly. The clustering results suggest that the CP vectors w.r.t the counts of simplets can be applied to further downstream tasks, such as SC clustering or classification.

7.5 Q4. Simplet counting under Restricted Access

In Section 5, we have proposed a random-walk-based method SCRW for restricted access scenarios where we can only access a portion of an input SC instead of the whole SC. In this section, we compare SCRW with SC3, showing the comparable trade-off between speed and counting accuracy and the high memory efficiency of SCRW. Note that, since SCRW operates with local information obtained from neighbor queries, we can expect memory efficiency compared to SC3. For fairer comparisons, we further introduce a more memory-efficient version of SC3, called SC3-E, which does not save the calculated probabilities used for sampling the next subtree (Line 5 of Algorithm 3) but calculates probability each time. Hence, SC3-E uses less mem-

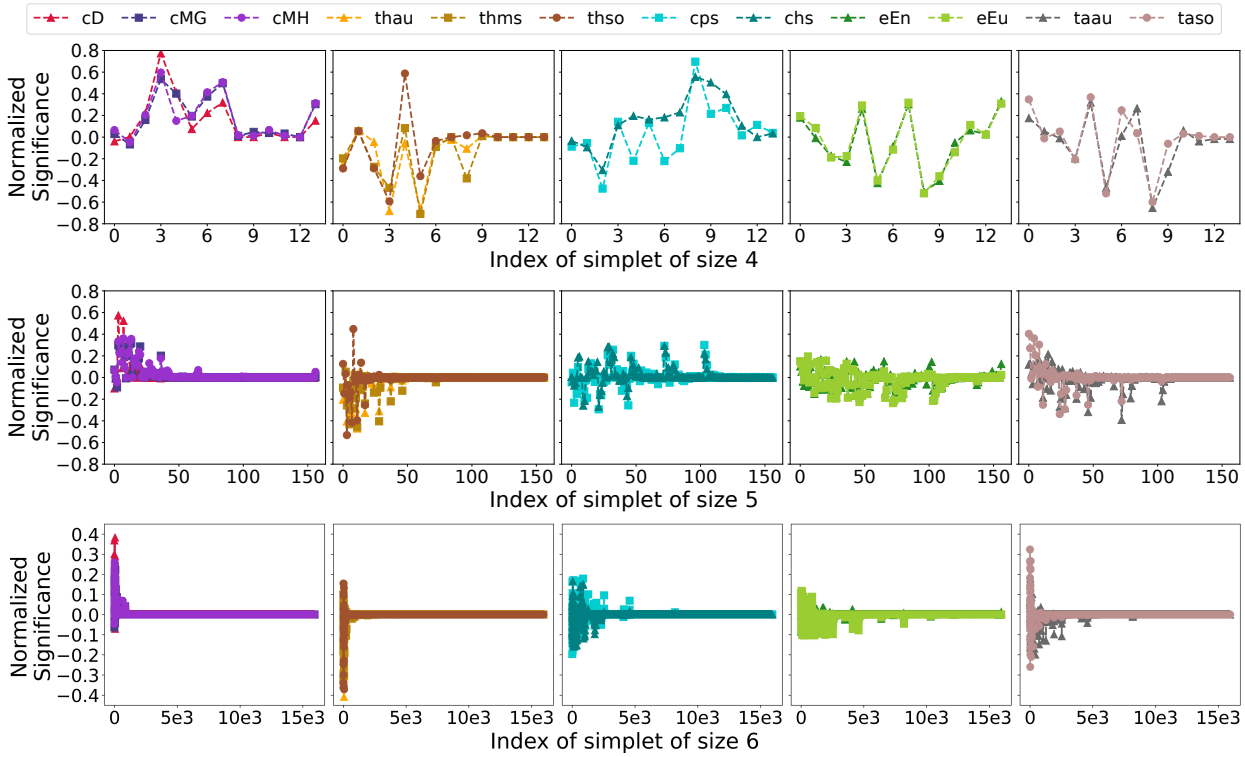


Fig. 8: The SC datasets from the same domain show similar CPs. We report the characteristic profile (CP) of each dataset based on the counts of simplets of sizes 4 (top), 5 (center), and 6 (bottom) estimated by SC3.

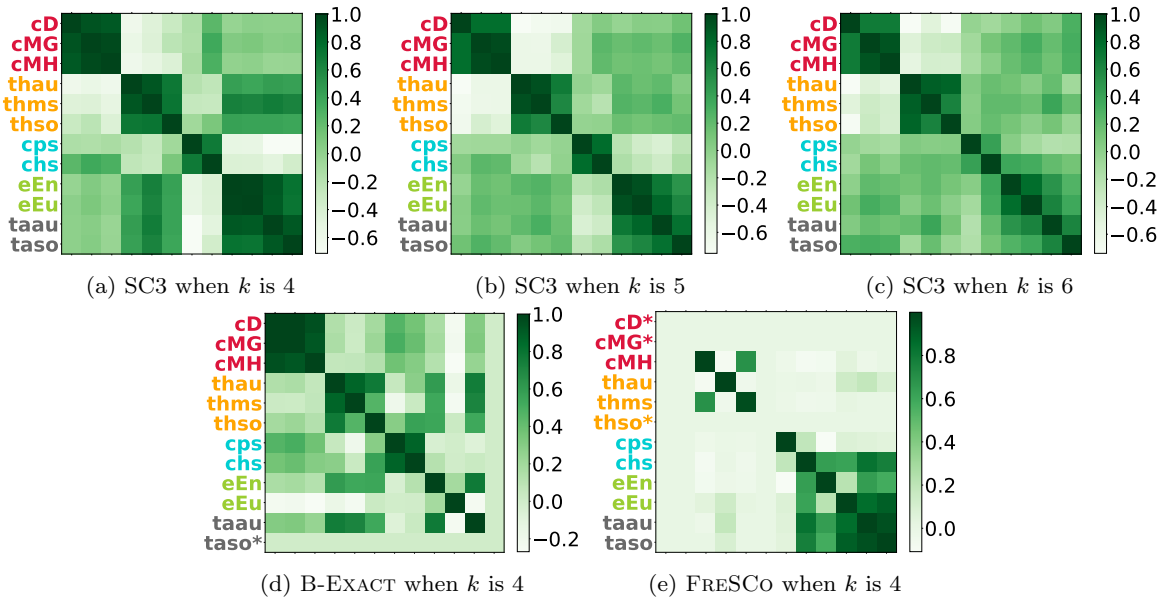
ory but requires more time than SC3.[24] Moreover, for comparing with SCRW, we modify the steps in SC3 and SC3-E as follows: scanning and matching are performed right after obtaining each sample, removing the need for memory storage for all samples, similar to SCRW. Additionally, to facilitate a direct comparison with SC3, neighbor queries are answered from the input SC loaded in the main memory, which is included in the memory usage of SCRW.

Speed and counting accuracy: We compare the accuracy of simplet counting and running time of different methods with the number of samples/queries $x \in \{10^2, 10^3, 10^4, 10^5\}$. For the largest `thso` dataset, we varied the number of queries for SCRW up to 10^6. As mentioned in Section 5.4, SCRW is limited to output the relative counts of simplets instead of counts and operate on the largest connected component (LCC) (refer to Appendix G for the analysis between LCC and the entire SC). When k is 4, we calculated the error between the relative counts of simplets of LCC from each method and the ground truth on the entire SC obtained by B-Exact. Specifically, we used the normalized error $\hat{err}_{\mathcal{K}}$ for accuracy comparison, which is defined as

$$\hat{err}_{\mathcal{K}} = \sum_{i \in [s_k]} |n_{oc}(\mathcal{S}_i^k; \mathcal{K}) - \tilde{n}_{oc}(\mathcal{S}_i^k; \mathcal{K})|.$$

Since B-Exact does not support $k \geq 5$, we cannot obtain the ground truth and utilize the normalized error $\hat{err}_{\mathcal{K}}$. Instead, we used the sum of standard deviations as an accuracy measure. Specifically, we calculated the standard deviation for each simplet over 20 trial outputs and then compared the overall accuracy using the sum of these standard deviations. Figure 10 shows the trade-off between speed and accuracy for each method on the two largest datasets, `taso` and `thso`. When comparing the running times of methods for achieving similar counting accuracies, the superior method varies depending on the dataset. Specifically, SCRW requires up to 3.6× less time to obtain similar counting accuracies on the `thso` datasets when k is 4, while SCRW requires up to 4.6× more time on the `taso` dataset when k is 4. On average, SCRW requires 4.7× less time on 5 datasets when k is 4, while it requires 9.5× more time on 11 datasets when k is 4. The relative superiority is largely determined by the pre-processing time required for the building step of SC3. The results and analysis for all datasets are provided in Appendix J [33]. Further-

[24] The worst-case time and space complexities of SC3-E and SC3 are equivalent, assuming all sampled trees are disjoint (see Appendix I for more details).



(a) SC3 when k is 4 (b) SC3 when k is 5 (c) SC3 when k is 6

(d) B-Exact when k is 4 (e) FreSCo when k is 4

Fig. 9: The domains of the datasets are better distinguished when the CPs are based on the outputs of SC3. For each considered algorithm, we report the similarity between the CPs of each pair of datasets. The detailed numerical results are in Table 4. An asterisk (*) indicates that B-Exact and FreSCo ran out of time ($\geq$ 10 hours) on the corresponding dataset.

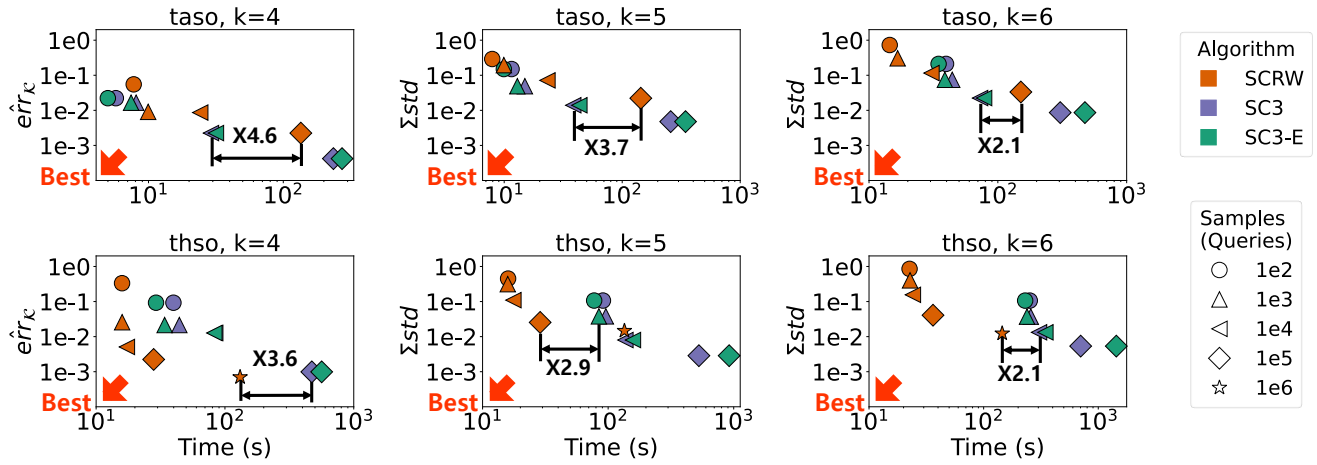


Fig. 10: SCRW and SC3 offer comparable speed-accuracy trade-offs. We report the trade-offs between speed and counting accuracy for each method across the `taso` and `thso` datasets when k is 4 (left), 5 (center), and 6 (right). We report the mean value of the measurement outputs and times over 20 trials on each dataset. When comparing the running times of methods for achieving similar counting accuracies, SCRW requires up to 3.6× less time than SC3 on the `thso` datasets, while SCRW requires up to 4.6× more time on the `taso` dataset.

more, the accuracy of simplet counting, when varying the number of queries y, is provided in Figure 3 (spec., the right subfigure). It shows that the counts obtained by SCRW approach the exact counts, as the number of queries increases.

Memory: We compare the memory usage of different methods with the number of samples/queries $x \in \{10^2, 10^3, 10^4, 10^5\}$. Figure 11 shows that, as the number of samples increases, the memory usage for SC3 increases, while the memory usages for SC3-E and SCRW remain consistent. Specifically, SC3, which stores the probabilities of a subtree to sample the next one, generally requires more memory space than SC3-E (refer to Appendix I). Also, while SC3-E requires memory for the building step, accessing the entire SC, SCRW does not, and thus SCRW requires the smallest amount

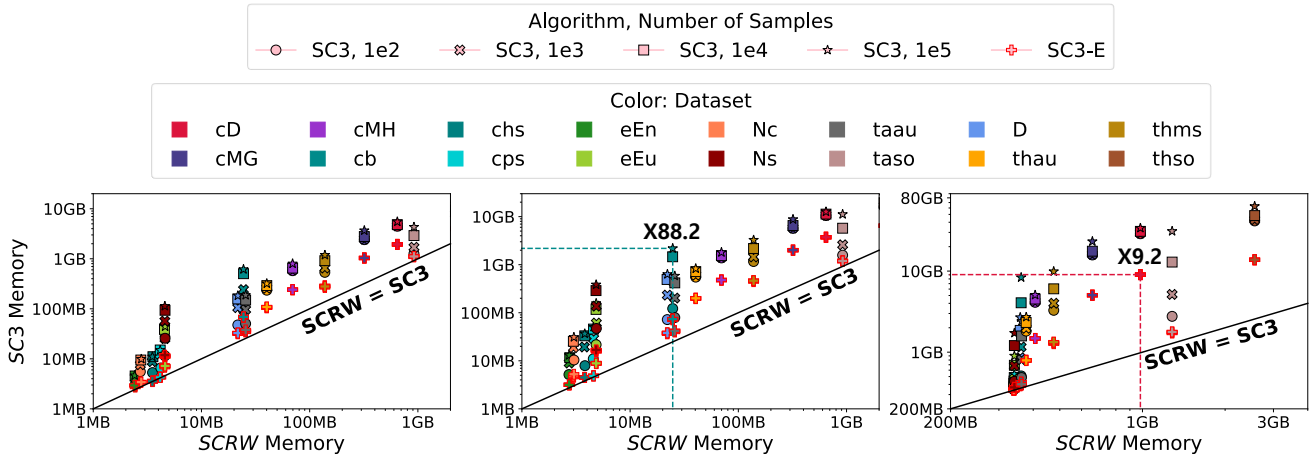


Fig. 11: SCRW is memory efficient. SC3 and SC3-E memory requirements of each dataset compared to SCRW as the number of samples/queries increases for k values of 4 (left), 5 (center), and 6 (right). We report the mean value of memory requirements over 5 trials. As the number of samples increases, the memory requirements for SC3 increase, while those for SC3-E and SCRW remain consistent. The straight line (SCRW = SC3), which corresponds to the points where the memory usage of both methods is equal, is added for easier comparison.

of memory (refer to Theorem 3 and 5). In particular, SCRW uses up to 88.2× less memory than SC3 (on the `cb` dataset) when k is 5 and, on average, it uses 16.5× less memory across 16 datasets. Furthermore, SCRW uses 9.2× less memory than SC3-E (on the `cD` dataset) when k is 6 and, on average, 2.6× less memory across 16 datasets.

8 Conclusions

In this section, we provide a summary of our work and discuss our contributions.

8.1 Summary of the Work

In this study, we tackle the challenge of counting simplets by introducing two sampling-based algorithms, SC3 and SCRW. We also show that the counts of simplets are effective in characterizing real-world simplicial complexes (SCs). Our contributions are summarized as follows:

- **New Problem.** To the best of our knowledge, we are the first to formulate and study the problem of directly counting simplets in a given SC, especially for the simplets beyond four nodes.
- **Accurate and Fast Algorithm.** SC3 is orders of magnitude faster than its competitors. Empirically, its running time is sub-linear w.r.t the number of samples. As a result, SC3 succeeds in estimating the count of every simplet of size 4, 5, and 6 in large SCs, and the result is accurate with theoretical guarantees.
- **Characterization of Real-world SCs.** We demonstrate that the output of SC3 can be used to characterize SCs. Especially, the characteristic profiles (CPs) based on the count of simplets of size 5 or 6 obtained by SC3 better distinguish the domains of real-world SCs than the CPs from its competitors.
- **Realistic scenarios.** We explore simplet counting under scenarios where the input SC is only partially accessible, introducing SCRW for such scenarios. Compared to SC3, SCRW is more memory-efficient, with a comparable trade-off between speed and counting accuracy.

For reproducibility, our code and data are available at `https://github.com/hhyy0401/simplet_counting`.

8.2 Discussion on Technical Contributions and Beyond

In this work, we address the novel problem of counting simplet occurrences. While the problem is not directly reducible to graphlet counting (in the primal graph) and our methods do have simplet-specific steps (e.g., simplet matching), as previously emphasized, ideas from graph counting methods (especially, CC [12–14] and SRW [15]) help address key technical challenges. We stand on the shoulders of giants in this regard.

Given this new problem, one of our non-trivial steps is to identify the appropriate building blocks that can be extended to this new problem without introducing unnecessary complexity. Note that, among a wide range

of (approximate) graphlet counting algorithms, not all can be adapted for simplet counting without significant additional complexity. In this respect, we believe that demonstrating how simplet counting can be done both effectively and efficiently using proper graphlet-related techniques as a foundation is a non-trivial contribution in itself. Another important contribution is the scalable (and open-sourced) implementation of these algorithms that significantly outperform existing methods on real-world datasets. We believe that turning ideas into concrete implementations and verifying their effectiveness adds substantial value to the research. Lastly, beyond the technical aspects, the strong characterization power of simplet counting and the novel discoveries gained from real-world datasets (e.g., domain-based similarity) highlight the practical significance of our work.

References

1. M. Agostini, M. Bressan, and S. Haddadan. Mixing time bounds for graphlet random walks. *Information Processing Letters*, 152:105851, 2019.
2. N. K. Ahmed, J. Neville, R. A. Rossi, and N. Duffield. Efficient graphlet counting for large networks. In *ICDM*, 2015.
3. N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995.
4. D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.
5. S. Barbarossa and S. Sardellitti. Topological signal processing over simplicial complexes. *IEEE Transactions on Signal Processing*, 68:2992–3007, 2020.
6. O. Ben-Eliezer, T. Eden, J. Oren, and D. Fotakis. Sampling multiple nodes in large networks: Beyond random walks. In *WSDM*, 2022.
7. A. R. Benson, R. Abebe, M. T. Schaub, A. Jadbabaie, and J. Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230, 2018.
8. S. K. Bera, J. Choudhari, S. Haddadan, and S. Ahmadian. Demetris: Counting (near)-cliques by crawling. In *WSDM*, 2023.
9. S. K. Bera and C. Seshadhri. How to count triangles, without seeing the whole graph. In *KDD*, 2020.
10. M. A. Bhuiyan, M. Rahman, M. Rahman, and M. Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *ICDM*, 2012.
11. G. Bianconi. *Higher-order networks*. Cambridge University Press, 2021.
12. M. Bressan, F. Chierichetti, R. Kumar, S. Leucci, and A. Panconesi. Counting graphlets: Space vs time. In *WSDM*, 2017.
13. M. Bressan, F. Chierichetti, R. Kumar, S. Leucci, and A. Panconesi. Motif counting beyond five nodes. *ACM Transactions on Knowledge Discovery from Data*, 12(4):1–25, 2018.
14. M. Bressan, S. Leucci, and A. Panconesi. Motivo: Fast motif counting via succinct color coding and adaptive sampling. *PVLDB*, 12(11):1651–1663, 2019.
15. X. Chen, Y. Li, P. Wang, and J. Lui. A general framework for estimating graphlet statistics via random walk. *PVLDB*, 10(3):253–264, 2016.
16. X. Chen and J. C. Lui. Mining graphlet counts in online social networks. *ACM Transactions on Knowledge Discovery from Data*, 12(4):1–38, 2018.
17. H. Cheng, X. Yan, and J. Han. Mining graph patterns. In *Frequent pattern mining*, pages 307–338. Springer, 2014.
18. N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on computing*, 14(1):210–223, 1985.
19. F. Chierichetti, A. Dasgupta, R. Kumar, S. Lattanzi, and T. Sarlós. On sampling nodes in a network. In *WWW*, 2016.
20. F. Chierichetti and S. Haddadan. On the Complexity of Sampling Vertices Uniformly from a Graph. In *ICALP*, 2018.
21. M. T. Do, S.-e. Yoon, B. Hooi, and K. Shin. Structural patterns and generative models of real-world hypergraphs. In *KDD*, 2020.
22. E. Estrada and G. J. Ross. Centralities in simplicial complexes. applications to protein interaction networks. *Journal of Theoretical Biology*, 438:46–60, 2018.
23. J. H. Fowler. Connecting the congress: A study of cosponsorship networks. *Political Analysis*, 14(4):456–487, 2006.
24. J. H. Fowler. Legislative cosponsorship networks in the us house and senate. *Social Networks*, 28(4):454–465, 2006.
25. O. Häggström. *Finite Markov chains and algorithmic applications*, volume 52. Cambridge University Press, 2002.
26. G. Han and H. Sethu. Waddling random walk: Fast and accurate mining of motif statistics in large graphs. In *ICDM*, 2016.
27. C. R. Harshaw, R. A. Bridges, M. D. Iannacone, J. W. Reed, and J. R. Goodall. Graphprints: Towards a graph analytic method for network anomaly detection. In *CISRC*, 2016.
28. T. Hočevar and J. Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.
29. I. Iacopini, G. Petri, A. Barrat, and V. Latora. Simplicial models of social contagion. *Nature Communications*, 10(1):1–9, 2019.
30. M. Jerrum and K. Meeks. The parameterised complexity of counting connected subgraphs and graph motifs. *Journal of Computer and System Sciences*, 81(4):702–716, 2015.
31. J. Jonsson. *Simplicial complexes of graphs*. Springer Science & Business Media, 2007.
32. H. Kim, J. Ko, F. Bu, and K. Shin. Characterization of simplicial complexes by counting simplets beyond four nodes. In *WWW*, 2023.
33. H. Kim, H. Moon, F. Bu, J. Ko, and K. Shin. Code, datasets, and appendix. `https://github.com/hhyy0401/simplet_counting`, 2024.
34. A. Knobbe, B. Crémilleux, J. Fürnkranz, and M. Scholz. From local patterns to global models: the lego approach to data mining. *LeGo*, 8:1–16, 2008.
35. G. Lee, J. Ko, and K. Shin. Hypergraph motifs: Concepts, algorithms, and discoveries. *PVLDB*, 13(11):2256–2269, 2020.
36. J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data*, 1(1):2–es, 2007.
37. Z. Li, Z. Deng, Z. Han, K. Alfaro-Bittner, B. Barzel, and S. Boccaletti. Contagion in simplicial complexes. *Chaos, Solitons & Fractals*, 152:111307, 2021.
38. Q. F. Lotito, F. Musciotto, A. Montresor, and F. Battiston. Higher-order motif analysis in hypergraphs. *Communications Physics*, 5(1):1–8, 2022.

39. D. Marcus and Y. Shavitt. Rage–a rapid graphlet enumerator for large networks. *Computer Networks*, 56(2):810–819, 2012.
40. R. Mastrandrea, J. Fournet, and A. Barrat. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PloS one*, 10(9):e0136497, 2015.
41. B. D. McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26(2):306–324, 1998.
42. T. Milenković, W. L. Ng, W. Hayes, and N. Pržulj. Optimal network alignment with graphlet degree vectors. *Cancer informatics*, 9:CIN–S4744, 2010.
43. J. Milnor. The geometric realization of a semi-simplicial complex. *Annals of Mathematics*, pages 357–362, 1957.
44. R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, 2004.
45. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.
46. A. Pinar, C. Seshadhri, and V. Vishal. Escape: Efficiently counting all 5-vertex subgraphs. In *WWW*, 2017.
47. G. Preti, G. De Francisci Morales, and F. Bonchi. Fresco: Mining frequent patterns in simplicial complexes. In *WWW*, 2022.
48. N. Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.
49. N. Pržulj, D. G. Corneil, and I. Jurisica. Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.
50. M. Rahman, M. A. Bhuiyan, and M. Al Hasan. Graft: An efficient graphlet counting method for large graph analysis. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2466–2478, 2014.
51. V. Salnikov, D. Cassese, and R. Lambiotte. Simplicial complexes and complex systems. *European Journal of Physics*, 40(1):014001, 2018.
52. M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie. Random walks on simplicial complexes and the normalized hodge 1-laplacian. *SIAM Review*, 62(2):353–391, 2020.
53. M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra. Signal processing on higher-order networks: Livin'on the edge... and beyond. *Signal Processing*, 187:108149, 2021.
54. N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *AISTATS*, 2009.
55. A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, and K. Wang. An overview of microsoft academic service (mas) and applications. In *WWW*, 2015.
56. G. M. Slota and K. Madduri. Fast approximate subgraph counting and enumeration. In *ICPP*, 2013.
57. J. Stehlé, N. Voirin, A. Barrat, C. Cattuto, L. Isella, J.-F. Pinton, M. Quaggiotto, W. Van den Broeck, C. Régis, B. Lina, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PloS one*, 6(8):e23176, 2011.
58. R. Street. The algebra of oriented simplexes. *Journal of Pure and Applied Algebra*, 49(3):283–335, 1987.
59. J. Tětek and M. Thorup. Edge sampling and graph parameter estimation via vertex neighborhood accesses. In *STOC*, 2022.
60. L. Torres, A. S. Blevins, D. Bassett, and T. Eliassi-Rad. The why, how, and when of representations for complex systems. *SIAM Review*, 63(3):435–485, 2021.
61. W. T. Tutte and W. T. Tutte. *Graph theory*, volume 21. Cambridge university press, 2001.
62. M. D. Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on software engineering*, 17(9):972–975, 1991.
63. D. Wang, Y. Zhao, H. Leng, and M. Small. A social communication model based on simplicial complexes. *Physics Letters A*, 384(35):126895, 2020.
64. P. Wang, J. C. Lui, B. Ribeiro, D. Towsley, J. Zhao, and X. Guan. Efficiently estimating motif statistics of large networks. *ACM Transactions on Knowledge Discovery from Data*, 9(2):1–27, 2014.
65. P. Wang, Y. Qi, J. C. Lui, D. Towsley, J. Zhao, and J. Tao. Inferring higher-order structure statistics of large networks from sampled edges. *IEEE Transactions on Knowledge and Data Engineering*, 31(1):61–74, 2017.
66. P. Wang, J. Zhao, X. Zhang, Z. Li, J. Cheng, J. C. Lui, D. Towsley, J. Tao, and X. Guan. Moss-5: A fast method of approximating counts of 5-node graphlets in large graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(1):73–86, 2017.
67. S. Wernicke and F. Rasche. Fanmod: a tool for fast network motif detection. *Bioinformatics*, 22(9):1152–1153, 2006.
68. C. Yang, M. Lyu, Y. Li, Q. Zhao, and Y. Xu. Ssrw: a scalable algorithm for estimating graphlet statistics based on random walk. In *DASFAA*, 2018.
69. D. M. Yellin. Algorithms for subset testing and finding maximal sets. In *SODA*, 1992.
70. H. Yin, A. R. Benson, J. Leskovec, and D. F. Gleich. Local higher-order graph clustering. In *KDD*, 2017.
71. J. Zhu, C. Wang, C. Gao, F. Zhang, Z. Wang, and X. Li. Community detection in graph: An embedding method. *IEEE Transactions on Network Science and Engineering*, 9(2):689–702, 2021.

A Proofs

Lemma 1 *Given primal graph $G_{\mathcal{K}} = (V_G, E_G)$, $k \in \mathbb{N}$, and a user-defined number x of samples, the building and sampling steps (Algorithms 2 and 3) output T_{ct} consisting of x node sets $V_k \in V_{con}$ in $O(c^k|E_G| + xk)$ time and $O(c^k|E_G|)$ space for some absolute constant $c > 1$, where $V_{con} = \{V_k \in \binom{V_G}{k} : G_{\mathcal{K}}[V_k]$ is connected$\}$. Moreover, in each iteration of sampling, a V_k is sampled with a probability proportional to $n_{st}(G_{\mathcal{K}}[V_k])$, the number of spanning trees of $G_{\mathcal{K}}[V_k]$.*

Proof Similar to the algorithms in [13], SC3-build (Algorithm 2) takes $O(c^k|E_G|)$ time and space and obtaining each sample using SC3-sample (Algorithm 3) takes $O(k)$ time and $O(c^k|E_G|)$ space (see the statement of Theorem 5.1 in [13]). The same complexities can be obtained since a subcomplex of an SC $\mathcal{K}$ induced by a node set V' is connected if and only if the induced subgraph of its primal graph on the same node set is connected (refer to Footnote 12 for a proof). Specifically, in SC3-build (Algorithm 2), for each $\mathcal{T}$ and each S, each $(u, v) \in E_G$ is considered twice (once for u and once for v), and there are $O(2^{|S|}) = O(2^k)$ ways to split S into S_1 and S_2. Also, the total numbers of possible $\mathcal{T}$'s and possible S's are both $O(2^k)$, which gives the time and space complexities $O(c_1^k|E_G|)$ for a constant $c_1 > 1$. In SC3-sample, in each

sampling, choosing v and $\mathcal{T}$ takes $O(1)$ time [62], each call of the function "Sample" also takes $O(1)$ time [62], and the function is called $O(|\mathcal{T}|) = O(k)$ times.

Lemma 2 *Given T_{ct} and $\mathcal{K} = (V, E)$, Algorithm 4 correctly outputs $M(\mathcal{K}[V_T])$ for all $V_T \in T_{ct}$ in $O(|M(\mathcal{K})|\hat{M}_{ct})$ time and $O(\hat{M}_{ct})$ space, where $\hat{M}_{ct} = \sum_{V_T \in T_{ct}} |M(\mathcal{K}[V_T])|$.*

Proof For each $V_T \in T_{ct}$, the number of σ's to check is $O(|M(\mathcal{K})|)$, and maintaining the set of all the maximal sets takes $O(|M(\mathcal{K})||M(\mathcal{K}[V_T])|)$ time [69]. The space complexity is $O(\sum_{V_T \in T_{ct}} |M(\mathcal{K}[V_T])|)$ since we output $M(\mathcal{K}[V_T])$ for each $V_T \in T_{ct}$, and the correctness immediately follows the definition of maximal simplices and how we update $M(\mathcal{K}[V_T])$ by only including maximal ones.

Lemma 3 *Given (i) k, the considered size of simplets, (ii) $\mathcal{S}^k$, derived from k, (iii) N_{ct}, obtained from the building step, (iv) T_{ct}, obtained from the scanning step, (v) $M(\mathcal{K}[V_T])$ ($\forall V_T \in T_{ct}$), obtained from the sampling step, Algorithm 5 takes $O(k!\hat{M}_k + |T_{ct}| + k^\omega)$ time and $O(\hat{M}_k + \hat{M}_{ct})$ space, where $\hat{M}_k = \sum_{\mathcal{S} \in \mathcal{S}^k} |M(\mathcal{S})|$, $\hat{M}_{ct} = \sum_{V_T \in T_{ct}} |M(\mathcal{K}[V_T])|$, and ω is the exponent of the time complexity of matrix multiplication. Here, $\hat{M}_k$ is a function of k: $\hat{M}_4$ is 47 and $\hat{M}_5$ is 807.*

Proof Building the permutation-invariant maps takes $O(|M(\mathcal{S})|)$ for each bijection ϕ and for each $\mathcal{S} \in \mathcal{S}^k$. The number of spanning trees for each $\mathcal{S} \in \mathcal{S}^k$ can be computed by using Kirchhoff's Matrix-Tree theorem, with a time complexity of $O(k^\omega)$, where ω is the matrix multiplication exponent. Enumerating the whole T_{ct} and accumulating the estimated counts takes $O(|T_{ct}|)$ times. The space complexity is $O(\sum_{\mathcal{S} \in \mathcal{S}^k} |M(\mathcal{S})| + \sum_{V_T \in T_{ct}} |M(\mathcal{K}[V_T])|)$ since we need $M(\mathcal{K}[V_T])$ for each $V_T \in T_{ct}$ and create $|M(\mathcal{S})|$ entries in f for each $\mathcal{S} \in \mathcal{S}^k$.

Theorem 1 (unbiasedness) *Given k, $\mathcal{K}$, $\mathcal{S}^k$, and any x, for each $\mathcal{S} \in \mathcal{S}^k$, the $\tilde{N}_{oc}(\mathcal{S})$ given by Algorithm 6 satisfies that $\mathbb{E}[\tilde{N}_{oc}(\mathcal{S})] = N_{oc}(\mathcal{S})$.*

Proof By Lemma 1, for each $\mathcal{S} \in \mathcal{S}^k$, let $V_\mathcal{S} \subseteq \binom{V}{k}$ with $|V_\mathcal{S}| = N_{oc}(\mathcal{S})$ denote the set of occurrences of $\mathcal{S}$. Each connected $V_T \in V_\mathcal{S}$ is colorful with probability $k!/k^k$, and at each iteration of sampling, V_T is sampled with probability $n_{st}(G_\mathcal{K}[V_T])/N_{ct}$. By the correctness of all the steps (Lemmas 1-3), the corresponding simplet of each sampled V_T is correctly found. Therefore, $\mathbb{E}[\tilde{N}_{oc}(S)] = \sum_{V_T \in V_\mathcal{S}} |T_{ct}| \frac{k!}{k^k} \frac{n_{st}(G_\mathcal{K}[V_T])}{N_{ct}} \frac{1}{n_{st}(G_\mathcal{K}[V_T])} \frac{N_{ct}}{|T_{ct}|} \frac{k^k}{k!} = \sum_{V_T \in V_\mathcal{S}} 1 = N_{oc}(\mathcal{S})$.

Theorem 2 (convergence) *Given any k, $\mathcal{K} = (V, E)$, and $\mathcal{S}^k$, for each $\mathcal{S} \in \mathcal{S}^k$, let $\tilde{N}^i_{oc}(\mathcal{S})$ denote the output by Algorithm 6 in the i-th trial. For any $\epsilon, \lambda > 0$, there exists $R_t = O(-\lambda^{-2}|V|^{2k} \ln \epsilon)$ such that if $R > R_t$, then $\Pr[|\sum_{i \in [R]} \tilde{N}^i_{oc}(\mathcal{S})/R - N_{oc}(\mathcal{S})| \leq \lambda] \geq 1 - \epsilon$, for any $x \geq 1$, where x is the number of samples in the sampling step. For a single trial, $\Pr[|\tilde{N}_{oc}(\mathcal{S}) - N_{oc}(\mathcal{S}) r_{color}| \leq \lambda\sigma] \geq 1 - \frac{1}{\lambda^2}$ where $\sigma^2 = Var[\tilde{N}_{oc}(\mathcal{S})] = O(\frac{1}{x})$ (in terms of x only), and r_{color} is the ratio between the actual count of occurrences of colorful treelets corresponding to $\mathcal{S}$ and the expected count.*

Proof It also relies on the correctness of all the steps (Lemmas 1-3). The statement for one trial follows Chebyshev's inequality with $\sigma^2 = Var[\tilde{N}_{oc}(\mathcal{S})] = \frac{N_{oc}(\mathcal{S})}{x} \left(\frac{N_{ct} \cdot k^k / k!}{n_{st}(\mathcal{S})} - 1 \right)$, where x is the number of samples during the sampling step. In addition, we have a Chebyshev bound (notations borrowed from Corollary 5.5 in [13]) of the probability that a uniformly-randomly-drawn colorful simplet corresponds to S: the probability is in $\mu_S \pm 2\epsilon/(1+\epsilon)$ with probability $1 - \Omega((x\epsilon)^{-2})$. The convergence is obtained from Hoeffding's inequality. Since each trial is i.i.d. and unbiased and each $\tilde{N}_{oc} = O(|V|^k)$, we have $\Pr[|\sum_{i \in [R]} \tilde{N}^i_{oc}(\mathcal{S})/R - N_{oc}(\mathcal{S})| \leq \lambda] \geq 1 - 2\exp(-2R\lambda^2 O(|V|^{-2k}))$. Hence, there exists $R = \Theta(-\lambda^{-2}|V|^{2k} \ln \epsilon)$ such that $2\exp(-2R\lambda^2 O(|V|^{-2k})) \leq \epsilon$.

Theorem 3 (complexities) *Given k, $\mathcal{K} = (V, E)$, $\mathcal{S}^k$, and the number of samples x, Algorithms 6 takes $O(c^k|E_G| + x|M(\mathcal{K})|^2 + k!\hat{M}_k)$ time and $O(c^k|E_G| + x|M(\mathcal{K})| + \hat{M}_k)$ space for some absolute constant $c > 1$, where $E_G = E \cap \binom{V}{2}$, and $\hat{M}_k = \sum_{\mathcal{S} \in \mathcal{S}^k} |M(\mathcal{S})|$ is a function of k.*

Proof By Lemmas 1-3, the total time complexity is $O(c^k|E_G| + xk + |M(\mathcal{K})|\hat{M}_{ct} + k!\hat{M}_k + x + k^\omega) = O(c^k|E_G| + x|M(\mathcal{K})|^2 + k!\hat{M}_k)$, where $E_G = E \cap \binom{V}{2}$, and we have used $\hat{M}_{ct} = O(x|M(\mathcal{K})|)$ and $k = O(M(\mathcal{K}))$. The total space complexity is $O(c^k|E_G| + \hat{M}_{ct} + \hat{M}_k + \hat{M}_{ct}) = O(c^k|E_G| + x|M(\mathcal{K})| + \hat{M}_k)$.[25]

Lemma 4 *For a given SC $\mathcal{K} = (V, E)$, the state space of SCRW corresponds to the set of 1-simplices in E, i.e., $\mathcal{X} = \{e \in E : |e| = 2\}$. The stationary distribution of each state $X \in \mathcal{X}$ is uniquely determined as $\frac{|\mathtt{Nb}(e_X)|}{M}$ (defined as Line 11). The transition probability from X to X' is $\frac{1}{|\mathtt{Nb}(e_X)|}$.*

Proof The transition probability from X to X' is $\frac{1}{|\mathrm{Nb}(e_X)|}$ since the next state is randomly and uniformly chosen from $\mathrm{Nb}(e_X)$. Using the fact that the stationary distribution uniquely exists, it is sufficient to verify that it satisfies $\pi P = \pi$, where P is the transition matrix. For the set E of all 1-simplices in $\mathcal{K}$, $\pi(e) \cdot P = \sum_{e' \in E} \pi(e') \cdot \mathtt{pb}(X_{e'}, X_e) = \frac{\sum_{e' \in E} |\{e \in \mathtt{Nb}(e')\}|}{M} = \frac{|\mathtt{Nb}(e)|}{M} = \pi(e)$.

Theorem 4 (unbiasedness and convergence) *Given k, $\mathcal{K}$, $\mathcal{S}^k$, for any $\mathcal{S} \in \mathcal{S}^k$ and $y > 0$, the $\tilde{n}_{oc}(\mathcal{S})$ given by Algorithm 7 satisfies $\mathbb{E}[\tilde{n}_{oc}(\mathcal{S})] = n_{oc}(\mathcal{S})$ and $Pr[|\tilde{n}_{oc}(\mathcal{S}) - n_{oc}(\mathcal{S})| \leq \lambda\sigma] \geq 1 - \frac{1}{\lambda^2}$ for any $\lambda > 0$, where $\sigma^2 = \mathrm{Var}[\tilde{n}_{oc}(\mathcal{S})]$.*

Proof Let $X_\mathcal{S}$ be a set whose elements are ordered sequences of 1-simplices isomorphic to a simplet $\mathcal{S} \in \mathcal{S}^k$ and $Z = \bigcup_{\mathcal{S} \in \mathcal{S}^k} X_\mathcal{S}$ be the total sample space. Recall that for any $\overrightarrow{V} \in Z$, $\tilde{p}(\overrightarrow{V})$ is the value defined in Line 16 on Algorithm 7, and M is the global coefficient that ensures the ratio $\tilde{p}(\overrightarrow{V})/M$ corresponds to the probability of V being sampled, whose correctness is ensured by Lemma 4. Using the fact $|X_\mathcal{S}| = \alpha_\mathcal{S} \cdot N_{oc}(\mathcal{S})$, for each $\mathcal{S} \in \mathcal{S}^k$, $\mathbb{E}[\tilde{n}_{oc}(S)]$ is $\sum_{\overrightarrow{V} \in X_\mathcal{S}} \frac{1}{\alpha_\mathcal{S} \tilde{p}(\overrightarrow{V})} \cdot \frac{\tilde{p}(\overrightarrow{V})}{M} = \frac{|X_\mathcal{S}|}{\alpha_\mathcal{S} \cdot M} = \frac{N_{oc}(\mathcal{S})}{M}$, divided by $\sum_{\mathcal{S}' \in \mathcal{S}^k} \sum_{\overrightarrow{V} \in X_{\mathcal{S}'}} \frac{1}{\alpha_{\mathcal{S}'} \tilde{p}(\overrightarrow{V})} \cdot \frac{\tilde{p}(\overrightarrow{V})}{M} = \frac{\sum_{\mathcal{S}' \in \mathcal{S}^k} N_{oc}(\mathcal{S}')}{M}$, resulting in $n_{oc}(\mathcal{S})$. Note that, since M is not used during the procedure, there is no need to compute it. The statement of convergence holds by Chebyshev's inequality: for any $\lambda > 0$,

[25] The terms can be simplified by using $\max(|E_G|, M(\mathcal{K})) \leq |E|$ and regarding x as a constant, which gives total time complexity $O(c^k|E| + |E|^2 + k!\hat{M}_k)$ and total space complexity $O(c^k|E| + k!\hat{M}_k)$.

Table 5: The outputs of B-Exact, FreSCo, and SC3 on the Ns dataset when k is 4. We run SC3 with x set to 100,000 samples and report the means over 5 trials. For FreSCo, we set the threshold θ to 1 to obtain the most accurate output, and set a time limit of 10 hours.

index of $\mathcal{S}^4$	B-Exact	SC3	FreSCo
0	-	953999597.1	2973
1	-	1481594205.7	3096
2	-	19876052.5	3024
3	528575876	527726043.0	2964
4	278014263	279769791.8	2955
5	68051012	68128876.7	2940
6	61794552	61601934.2	2935
7	17027710	17451818.3	2917
8	7538854	7648915.7	2908
9	10806231	10907215.3	2903
10	6040967	6102560.0	2896
11	1577597	167242.9	2885
12	168752	139459.2	2885
13	-	3290324.8	2885

Algorithm 8: Simplet expansion from graphlets

Input: (1) k: the considered size of graphlets and simplets
(2) $\mathcal{G}^k$: the set of all the graphlets of size k
Output: $\mathcal{S}_k$: the set of all the simplets of size k

$\mathcal{S}_k \leftarrow \emptyset$ ▷ Initialization
Function Expand(i, C, S)**:**
  if $i \leq 2$ **then**
    $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{S\}$
    return
  $T \leftarrow \binom{C}{i} \setminus S$; $Q \leftarrow \emptyset$
  foreach $\Sigma \subseteq T$ **do**
    $S' \leftarrow S \cup \left(\bigcup_{I \in \Sigma} 2^I\right)$
    if $\nexists q \in Q$ *s.t.* $S' \simeq q$ **then**
      $Q \leftarrow Q \cup \{S'\}$
      Expand$(i-1, C, S')$
for $\mathcal{G} = (V_\mathcal{G} = [k], E_\mathcal{G}) \in \mathcal{G}^k$ **do**
  $C_\mathcal{G} \leftarrow \{C \subseteq [k] : |C| > 2 \wedge \binom{C}{2} \subseteq E_\mathcal{G}\}$ ▷ Cliques
  Expand$(k, C_\mathcal{G}, \{\emptyset\} \cup \binom{[k]}{1} \cup E_\mathcal{G})$
return S_k

where $\bar{X}$ (which corresponds to $\tilde{n}_{oc}(\mathcal{S})$ in our statement) is the sample mean of i.i.d. random variables $X_1 = X, \ldots, X_n$, $\mu = \mathbb{E}[X]$, and $\Sigma^2 = \mathrm{Var}[X]$, since $\Pr\left[|\bar{X} - \mu| \leq \frac{\lambda\Sigma}{\sqrt{n}}\right] \geq 1 - \frac{1}{\lambda^2}$ can be transformed into $\Pr\left[|\bar{X} - \mu| \leq \lambda'\Sigma\right] \geq 1 - \frac{1}{n\lambda'^2}$ under $\lambda' = \frac{\lambda}{\sqrt{n}}$, the right-hand term approaches 1 as $n \to \infty$, i.e., $\bar{X}$ converges to μ w.h.p.

Theorem 5 (complexities) *The space complexity of SCRW is $O(\mathcal{Q}_{\max} + \hat{M}_k)$, where $\mathcal{Q}_{\max} = \max_{v \in V} \sum_{\sigma \in \mathcal{Q}(v)} |\sigma|$ is determined by $\mathcal{K}$, and $\hat{M}_k = \sum_{\mathcal{S} \in \mathcal{S}^k} |M(\mathcal{S})|$ is a function of k. The time complexity of SCRW is $O(y|M(\mathcal{K})|^2 + k!\hat{M}_k + yk + y\mathcal{Q}_{\max})$, where y is the maximum number of queries used in Algorithm 7 under the assumption each neighborhood query with query node $v \in V$ takes $O(\sum_{\sigma \in \mathcal{Q}(v)} |\sigma|)$ time.*

Proof We can store each $\mathrm{pb}(\cdot,\cdot)$ and $|\mathrm{Nb}(\cdot)|$, $\mathrm{Nb}(\cdot)$, and $\tilde{\mathcal{Q}}$ for $k-2$, $k-2$, 1, and 1 steps after calculation, respectively, to obtain $k-1$ consecutive states, then discard them immediately thereafter to minimize memory usage. This requires $O(k + \mathcal{Q}_{\max})$ space, with an additional $O(\hat{M}_k)$ space for matching. Also, let $\hat{M}_{rw} = \sum_{V_S \in V_{rw}} |\mathcal{K}[V_S]|$ where V_{rw} is the set of sampled V_S's while running Algorithm 7. Then for time complexity, $O(|M(\mathcal{K})|\hat{M}_{rw})$ time and $O(k!\hat{M}_k + y)$ time are required in the scanning and matching step, respectively (refer to Lemma 2 and 3). For obtaining $\mathtt{Nb}(\cdot)$ and $\mathtt{pb}(\cdot)$ at each state, SCRW takes $O(\mathcal{Q}_{\max})$ time and $O(k)$ time, respectively. Therefore, the total time complexity is $O(|M(\mathcal{K})|\hat{M}_{rw} + k!\hat{M}_k + y + y\mathcal{Q}_{\max} + yk) = O(y|M(\mathcal{K})|^2 + k!\hat{M}_k + yk + y\mathcal{Q}_{\max})$, where $\hat{M}_{rw} = O(y|M(\mathcal{K})|)$.

B Baseline algorithms

B-Exact: For each size-4 simplet, we show which 4-node configurations (see Table 7 in the appendix of [7] for the details), if any, are corresponding to it: (1) $\mathcal{S}_4^0 : \emptyset$; (2) $\mathcal{S}_4^1 : \emptyset$; (3) $\mathcal{S}_4^2 : \emptyset$; (4) $\mathcal{S}_4^3 : \psi_0$; (5) $\mathcal{S}_4^4 : \psi_1, \psi_2$; (6) $\mathcal{S}_4^5 : \theta_{0,0}$; (7) $\mathcal{S}_4^6 : \theta_{0,1}, \theta_{0,2}$; (8) $\mathcal{S}_4^7 : \theta_{1,1}, \theta_{1,2}, \theta_{2,2}$; (9) $\mathcal{S}_4^8 : \pi_{0,0,0,0}$; (10) $\mathcal{S}_4^9 : \pi_{0,0,0,1}, \pi_{0,0,0,2}$; (11) $\mathcal{S}_4^{10} : \pi_{0,0,1,1}, \pi_{0,0,1,2}, \pi_{0,0,2,2}$; (12) $\mathcal{S}_4^{11} : \pi_{0,1,1,1}, \pi_{0,1,1,2}, \pi_{0,1,2,2}, \pi_{0,2,2,2}$; (13) $\mathcal{S}_4^{12} : \pi_{1,1,1,1}, \pi_{1,1,1,2}, \pi_{1,1,2,2}, \pi_{1,2,2,2}, \pi_{2,2,2,2}$; (14) $\mathcal{S}_4^{13} : \emptyset$. We can see that the configurations in [7] and the simplets are not one-to-one corresponded.

FreSCo: We provide the definition of *support*, the surrogate measure used in [47], which is essentially different from the count of simplets *per se*, both theoretically and empirically.

Definition 2 (support [47]) Let the image set $I_P(v)$ of $v \in V_P$ of a simplet P be the set of nodes in $\mathcal{K}$ that are mapped to v by some isomorphism ϕ, i.e., $I_P(v) = \{u \in \mathcal{K} : \exists\phi \text{ s.t. } \phi(u) = v\}$. Then the support of P in $\mathcal{K}$ is $SUP(P, \mathcal{K}) = \min_{v \in V_P} |I_P(v)|$.

Roughly speaking, the occurrence of a simplet (which our algorithms count) is the count of subcomplex-level mappings, while the support of a simplet is the minimum aggregation of the counts of node-level mappings. To clarify their distinction, we provide examples where they differ and where they are the same as following. Consider counting $\mathcal{S}_{13}^4$ (refer to Fig. 2) within $\mathcal{S}_{13}^4$ itself. While the occurrence is 1, the support is 4 since any node can be mapped with any of the four nodes due to the symmetry of the topology. However, when counting $\mathcal{S}_6^4$ within $\mathcal{S}_6^4$ itself, both the occurrence and support are 1, as each node can only be mapped to one specific node, due to the lack of symmetry. Despite this potential distinction, FreSCo uses support because it satisfies the anti-monotone property, which FreSCo exploits for efficiency. Specifically, the support of a simplet is always at most that of its sub-simplets.

In Table 5, when k is 4, we compare the outputs of B-Exact,[26] FreSCo, and SC3, where we can see that the output of SC3 is an accurate estimator of the exact count of simplets, while the output of FreSCo is hardly meaningful.

[26] For B-Exact, for each simplet, we sum up the counts of the node configurations corresponding to the simplet.

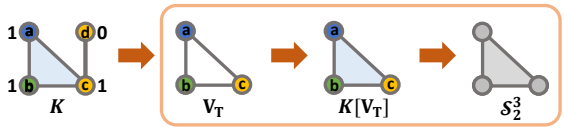


Fig. 12: The subfigures correspond to the building, sampling, scanning, and matching steps in order.

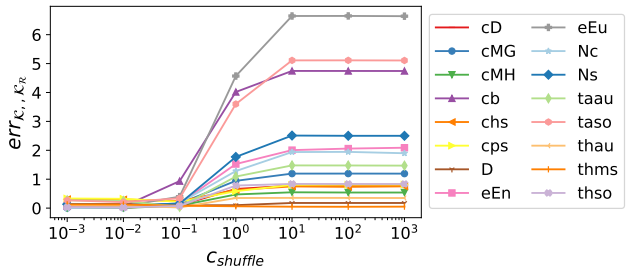


Fig. 13: The error nearly converges when $c_{shuffle} \geq 10$. This indicates that a $c_{shuffle}$ value of 1,000 is sufficient to randomize the considered SCs.

C Simplet expansion from graphlets

In Algorithm 8, we provide a practical way to generate the set of all possible simplets of size k. Essentially, we conduct an expansion from all the graphlets of size k to all the simplets of size k. We assume that all the graphlets of size k are known beforehand,[27] and use them as the input of SC3. Specifically, we exploit the fact that, for each k, there is a surjection from all the simplets of size k to all the graphlets of size k, where each simplet is mapped to its primal graph.

We first save all the cliques of size more than 2 for each graphlet $\mathcal{G} \in \mathcal{G}^k$ (line 13), the candidates to be open or closed (i.e., covered with a simplex of the same size). Then, starting from cliques of possible maximum size (line 14) to edges (line 3), we consider the number of all cases ($\Sigma \in 2^T$ on line 7) that a clique of the current size i any of whose superset is not closed (T on line 6) is either open or closed recursively. If S' made by closing every $\Sigma \in 2^T$ is not isomorphic $\forall q \in Q$ (line 9), then we put S' into a simplet set Q (line 10) and keep expansion until there is no clique to be closed. Finally, when reaching the terminate condition, we put the simplet S to the set of simplets $\mathcal{S}_k$ (line 4).

D A toy example for SC3

See Figure 12 for the entire procedure. For a given SC of order 4 when k is 3, we aim to count $N_{oc}(\mathcal{S}_i^3; \mathcal{K})$ for each $i \in [s_3]$. In the building step, we count the number of colorful treelets rooted on each node (in **bold** next to each node) after coloring every node with three colors uniformly at random. In the sampling step, $V_T = \{a, b, c\}$ is sampled with probability 1, the only case of a size-3 colorful tree. In the scanning step, an induced subcomplex on V_T is found and matched to an isomorphic simplet ($\mathcal{S}_2^3$) in the final step. The three processes except for the building step are repeated.

[27] Technically, they can be generated via isomorph-free exhaustive generation [41].

E Effect of the shuffling ratio $c_{shuffle}$ in the null model on randomization

To generate a randomized SC $\mathcal{K}_\mathcal{R}$, we repeatedly shuffled the pair of maximal simplices $c_{shuffle}|E|$ times. We compare the counts of a subset of simplets countable by B-Exact between real SCs and randomized SCs, while varying the value of $c_{shuffle}$ from 0.001 to 1,000. For the error measure, we used a normalized error $err_{\mathcal{K},\mathcal{K}_\mathcal{R}}$ defined as

$$err_{\mathcal{K},\mathcal{K}_\mathcal{R}} = \frac{\sum_{i\in[s_k]} |N_{oc}(\mathcal{S}_i^k; \mathcal{K}) - N_{oc}(\mathcal{S}_i^k; \mathcal{K}_\mathcal{R})|}{\sum_{i\in[s_k]} N_{oc}(\mathcal{S}_i^k; \mathcal{K})},$$

where $N_{oc}(\mathcal{S}_i^k; \mathcal{K}_\mathcal{R})$ is the count from the randomized SC. Figure 13 shows that the difference between the counts from real SC and randomized SC increases as the value of $c_{shuffle}$ increases. The error, $err_{\mathcal{K},\mathcal{K}_\mathcal{R}}$, nearly converges once $c_{shuffle}$ exceeds 1,000, indicating that setting $c_{shuffle}$ to 1,000 is sufficient for effectively randomizing the considered SCs.

F Increases in running time with respect to data size

To address the potential bias arising from the use of different programming languages in baseline implementations, we compare the growth rates of their running times relative to that of dataset sizes, which is independent of absolute running times. Specifically, for each method and each dataset, we measured the relative growth rate as follows:

$$\overline{Time} = \frac{Time/Time_0}{|M(\mathcal{K})|/|M(\mathcal{K}_0)|}$$

where $Time$ and $|M(\mathcal{K})|$ represent the running time of the considered method and the size of the considered dataset; and $Time_0$ and $|M(\mathcal{K}_0)|$ denote the running time of the method on the smallest dataset (i.e., `eEn`) and its size. In particular, when comparing the running times of the SC3 algorithm, we use the running time when x is 100,000 as $Time_0$. Figure 14 shows that, the relative growth rate of SC3 is lower than that of the baseline methods in most cases, demonstrating the scalability of SC3.

G Simplet counts in largest connected components

We compare the exact simplet counts in the largest connected component (LCC) and the entire simplicial complex (SC) computed by B-Exact using the normalized difference $err_\mathcal{K}$ defined as

$$err_\mathcal{K} = \frac{\sum_{i\in[s_k]} |N_{oc}(\mathcal{S}_i^k; \mathcal{K}) - \bar{N}_{oc}(\mathcal{S}_i^k; \mathcal{K})|}{\sum_{i\in[s_k]} N_{oc}(\mathcal{S}_i^k; \mathcal{K})},$$

where $\bar{N}_{oc}(\mathcal{S}_i^k; \mathcal{K})$ is the count of simplet $\mathcal{S}_i^k$ in the LCC of SC $\mathcal{K}$. Figure 15 shows that the normalized difference $err_\mathcal{K}$ between LCC and the entire SC does not exceed 0.01 across all datasets. Especially, on 9 out of 16 datasets, both simplet counts are exactly the same. Refer to Table 3 for basic statistics of the LCC of each data.

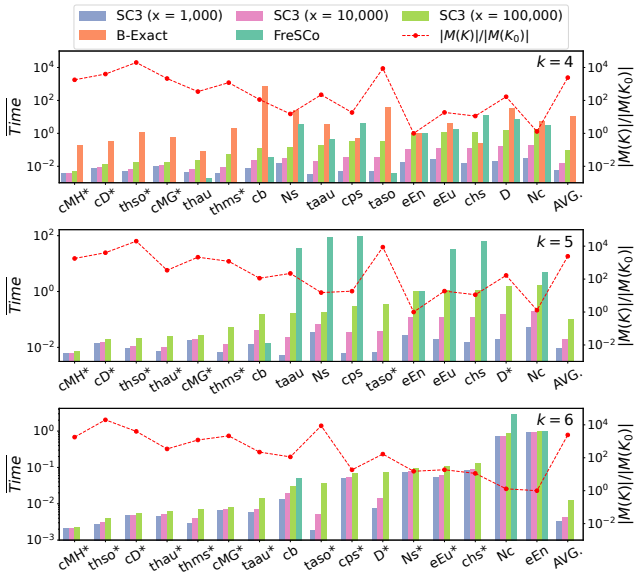


Fig. 14: SC3 exhibits smaller increase rates in running time relative to the growth in dataset size, compared to the baseline algorithms, in most cases. The red dotted lines indicate the relative dataset sizes. FreSCo ran out of time (> 10 hours) for some datasets, and B-Exact can be used only for $k \leq 4$.

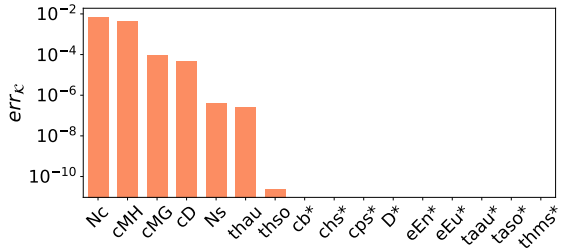


Fig. 15: The simplet count differences between the LCC and the entire SC are negligible. We report the normalized difference $err_{\mathcal{K}}$ between the exact counts of the LCC and the entire SC on each dataset. The difference is lower than 0.01 across all datasets, and both simplet counts are identical on the 9 datasets marked with *.

H Rationale for the selection of time budget τ

Recall that we used a pre-defined value of τ to sample subcomplexes after τ steps. In this experiment, we examined how varying the number of random walk steps (i.e., $y - 2$) from 1 to 50 affects the accuracy of SCRW while fixing τ to 0. As in previous experiments, we compare the ground truth counts computed by B-Exact and the estimated counts obtained by SCRW, using the normalized error $\hat{err}_{\mathcal{K}}$. Additionally, we compare the $\Delta\hat{err}_{\mathcal{K},z_i} = \hat{err}_{\mathcal{K},z_i} - \hat{err}_{\mathcal{K},z_{i-1}}$ to show the reduction in error compared to the previous number of steps, where $\hat{err}_{\mathcal{K},z_i}$ represents the $\hat{err}_{\mathcal{K}}$ when the number of steps is z_i. As seen in Figure 16, the error in SCRW tends to decrease as the number of steps increases, but the reduction in

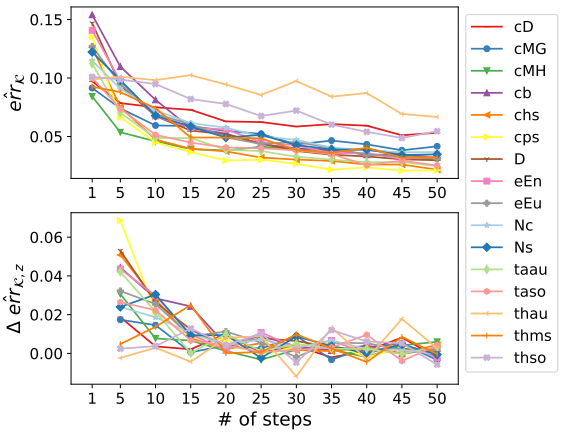


Fig. 16: The error in SCRW decreases more significantly within the first 20 steps, compared to later steps. We report the normalized error $err_{\mathcal{K}}$ and its difference $\Delta\hat{err}_{\mathcal{K},z}$. Notably, the normalized error $err_{\mathcal{K}}$ significantly decreases at 20 steps across all datasets.

$\hat{err}_{\mathcal{K}}$ is particularly significant within the first 20 steps, compared to later steps. These results justify our choice of setting the time budget τ to 20 (i.e., disregarding the samples from the first 20 steps) for running SCRW.

I Memory efficient version of SC3 (SC3-E)

SC3 is designed to store the probabilities used to sample the next subtree (see Line 5 of Algorithm 3). Ideally, its memory usage should remain consistent when the same subtrees are sampled repeatedly. However, in practical scenarios, memory usage tends to increase each time new probabilities are stored. SC3-E does not save the calculated probabilities, ensuring a consistent memory requirement. The time complexity of SC3-E is equivalent to the worst-case scenario of SC3, where all sampled subtrees are disjoint. For space complexity, the size of distributions of all subtrees (see Line 5 of Algorithm 3) is bounded by $O(c^k|E_G|)$, making their complexities equivalent. See Theorem 3 for the time and space complexity of SC3.